TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 1802T007 – Informační technologie

Statistika WWW stránek

WWW pages statistic

Diplomová práce

Autor: **Bc. Jan Černý**
Vedoucí práce: Mgr. Jiří Vraný, Ph. D.

V Liberci 15. 5. 2013

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum

Podpis

Poděkování

Na tomto místě bych chtěl poděkovat mému vedoucímu diplomové práce, Mgr. Jiřímu Vranému, Ph.D., za vedení mé diplomové práce, který mi svými radami a konzultacemi pomohl nasměrovat práci ke zdárnému konci. V neposlední řadě bych chtěl poděkovat své rodině za vytvoření potřebného zázemí a poskytnutí finančních prostředků ke studiu. Také bych chtěl poděkovat za podporu ze strany mých přátel a známých.

Abstrakt

Diplomová práce se zabývá problematikou automatického získávání dat z WWW. Návrhu a implementaci vlastního řešení v podobě funkční aplikace pro získávání a statistické zpracování informací o WWW stránkách předchází stručná rešerše nástrojů pro automatické sbírání dat z WWW. Rešerše dále obsahuje podobné projekty, které již existují. Součástí aplikace jsou moduly pro nejdůležitější informace o stránce: na jakém serveru „běží“, jaká skriptovací technologie je použita, jaká je verze jazyka HTML a CSS. Zda je použit některý JavaScript framework. V závěru práce je předložen výsledek výzkumu – informace získané analýzou 1 200 035 unikátních World Wide Web stránek.

Klíčová slova: aplikace, Python, WWW, databáze, statistiky, stránka, vlákna, CouchDB, sPYnet, spycouch

Abstract

This diploma thesis deals with the issue of automatic WWW data mining. The actual design and implementation of the application for gathering and statistical analysis of web page information was preceded by a brief research of tools for automatic web data mining. The research also contains other similar projects that already exist. The application contains modules for the most important web page information: which server it runs on, which scripting technology is being used, which is the version of HTML and CSS, and if a JavaScript framework has been used. At the very end of the work, the reader is presented with the outcome of the research – information gathered from an analysis of 1,200,035 unique web pages.

Keywords: application, Python, WWW, database, statistic, page, threads, CouchDB, sPYnet, spycouch

Obsah

Seznam obrázků a tabulek

1 Úvod

Webové aplikace, které poskytují svůj obsah, si o svém klientovi získávají a většinou i zaznamenávají nejrůznější dostupné informace. Počínaje IP adresou, ze které přišel dotaz na server, přes používaný webový prohlížeč, operační systém, konče uživatelovými dotazy na vyhledávání nebo historií toho, co se mu líbí.

Z tohoto důvodu vznikla opačná myšlenka, která je tématem této diplomové práce, a to získávat a zpracovávat informace o WWW stránkách a takto nabyté údaje poskytnout formou ucelených statistik. Tyto statistiky budou veřejně přístupné na daném portálu a budou sloužit všem, kdo do nich budou chtít nahlédnout. Cílovou skupinou mohou být například začínající programátoři WWW aplikací, kterým poslouží k výběru toho nejběžněji používaného JavaScript frameworku a podobně.

Prvním cílem této diplomové práce je zmapovat, zda něco podobného již existuje a pokud ano, tak na základě této rešerše sestavit návrh vlastní aplikace pro automatické získávání dat a statistické zpracování informací o WWW stránkách. V pořadí druhým cílem je, aby bylo při návrhu aplikace dbáno na její modularitu, možnost jednotlivé sledované položky statistiky odebírat, nebo přidávat. Dalším cílem je dle vlastního návrhu vytvořit aplikaci, k jejíž implementaci má být použit programovací jazyk Python. Závěrečným cílem je v praxi otestovat její funkčnost a zanalyzovat pomocí ní, alespoň 1 milion WWW stránek a uvést výsledky analýzy.

2 Teoretická část

V této první větší ucelené části technické zprávy budou rozebrány teoretické základy, které by měly posloužit k vhledu do dané problematiky. Dále bude pojednáno o použitých nástrojích a v závěru kapitoly bude uvedena stručná rešerše podobných projektů.

2.1 Protokol HTTP

HTTP (Hypertext Transfer Protokol) je internetovým protokolem na aplikační úrovni. Od roku 1990 se používá pro výměnu hypertextových dokumentů v rámci Internetu. V současné době je nejrozšířenější ve své verzi HTTP/1.1, která je definovaná v RFC 2616[1]. Tato verze umožňuje přenášet jakékoliv soubory definované rozšířením MIME *(Multipurpose Internet Mail Extensions)*, ne jen hypertextové dokumenty.

Způsob fungování protokolu je dotaz-odpověď. Na jeden dotaz klienta je odeslána jedna odpověď. Na každý další dotaz je vždy odeslána nová, nezávislá na předchozím dotazu, proto je HTTP protokol nazýván bezstavovým protokolem.

Běžnému uživateli je komunikace se serverem skryta, ale v mnoha případech, jako například ke zjištění základních informací o serveru, které budou součástí statistik v této diplomové práci, je třeba se v ní lehce zorientovat. Proto je na jednoduchém příkladu demonstrováno, co se ve skutečnosti na pozadí komunikace odehrává. Mějme na serveru dokument *statistiky.html*, o který klient žádá.

Klientský dotaz:

```
GET /diplomka/statistiky.html HTTP/1.1
Host: www.cernyjan.cz
```

Serverová odpověď:

```
HTTP/1.1 200 OK
Date: Tue, 26 Mar 2013 20:38:42 GMT
Server: Apache
Last-Modified: Mon, 25 Mar 2013 18:07:15 GMT
ETag: "1b5167b-254-4d8c3acc9a51d"
Accept-Ranges: bytes
Content-Length: 596
Vary: Accept-Encoding,User-Agent
Connection: close
Content-Type: text/html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="cs"
lang="cs">
<head>
<meta http-equiv="content-type" content="text/html;
charset=utf-8" />
<meta http-equiv="refresh" content="1;url=statistiky.php">
<title>DP - statistiky</title>
<meta name="keywords" content="" />
<meta name="description" content="" />
<link rel="stylesheet" type="text/css" href="/default.css"
media="screen" />
<link rel="icon" type="image/png" href="/favicon.png" />
</head>
<body>
</body>
</html>
```

Z úvodu odpovědi lze vyčíst, že dokument je na serveru skutečně uložen *(200 OK)*, dále se klient dozvídá používanou verzi HTTP protokolu *(HTTP/1.1)*. Dalšími informacemi jsou, kdy byl dokument vytvořen, respektive naposledy modifikován, jeho velikost a podobně. Po úvodní části odpovědi – hlavičce, kterou by klient rovněž obdržel, pokud by GET nahradil za HEAD v dotazu, následuje celé tělo dokumentu *statistiky.html*, o který byl server požádán.

V příkladu byla použita dotazovací metoda GET, která je tou nejběžnější metodou, ale pro komunikaci se používají i další:

- **OPTIONS** – dotaz na schopnosti a omezení serveru

- **GET** – metoda již zmíněná v příkladu je spojena s požadavkem na zaslání dokumentu, na který byl server dotázán pomocí URL

- **HEAD** – obdobná s metodou GET, ale místo odpovědi obsahující celé tělo dokumentu, jsou klientovi zaslány pouze doplňkové informace o dokumentu (jeho velikost, znaková sada, …), toho se využívá například pro test, zda se dokument na serveru nachází, aniž by byl celý přenášen

- **POST** – pokud z nějakého důvodu není vhodné přenášet data viditelnou formou jako součást URL, například z hlediska bezpečnosti, používá se tato metoda, která slouží pro příjem dat serverem z požadavku (odesílání webových formulářů, poslání e-mailu, …)

- **PUT** – slouží pro uložení dat na server, běžně se k tomu využívá jiných služeb, například FTP, ale metoda PUT je využívána například při ukládání dokumentů do CouchDB databáze přes REST API, o které bude v další kapitole pojednáno

- **DELETE** – poslední metoda ze serveru smaže daný dokument

Jak je z příkladu patrné, součástí odpovědi je vždy i jeden ze stavových kódů. Tento stavový kód udává, jak byl dotaz serverem zpracován. Kód bývá doplněn o anglický popis pro lepší orientaci. Číselné kódy jsou rozděleny do pěti kategorií, dle první číslice v trojčíslí – *1xx, 2xx, 3xx, 4xx* a *5xx*. Číselné kódy, které bude třeba v aplikaci rozlišovat:

- **200 OK** – vše proběhlo bez chyby, server zasílá odpověď

- **301 Moved Permanently** – dokument byl trvale přesunut na jiné umístění, pokud ho klient chce získat, musí zadávat do dotazu novou URL, nové umístění je sdělováno v hlavičce parametrem *Location*

- **302 Moved Temporarily** – obdobné jako předchozí, jen se jedná o dočasné přemístění

- **404 Not Found** – dotaz na neexistující objekt na serveru, chyba vzniklá „překlepem“ v URL, nebo smazáním dokumentu v době před dotazem, o který klient žádá

2.2 Webová stránka

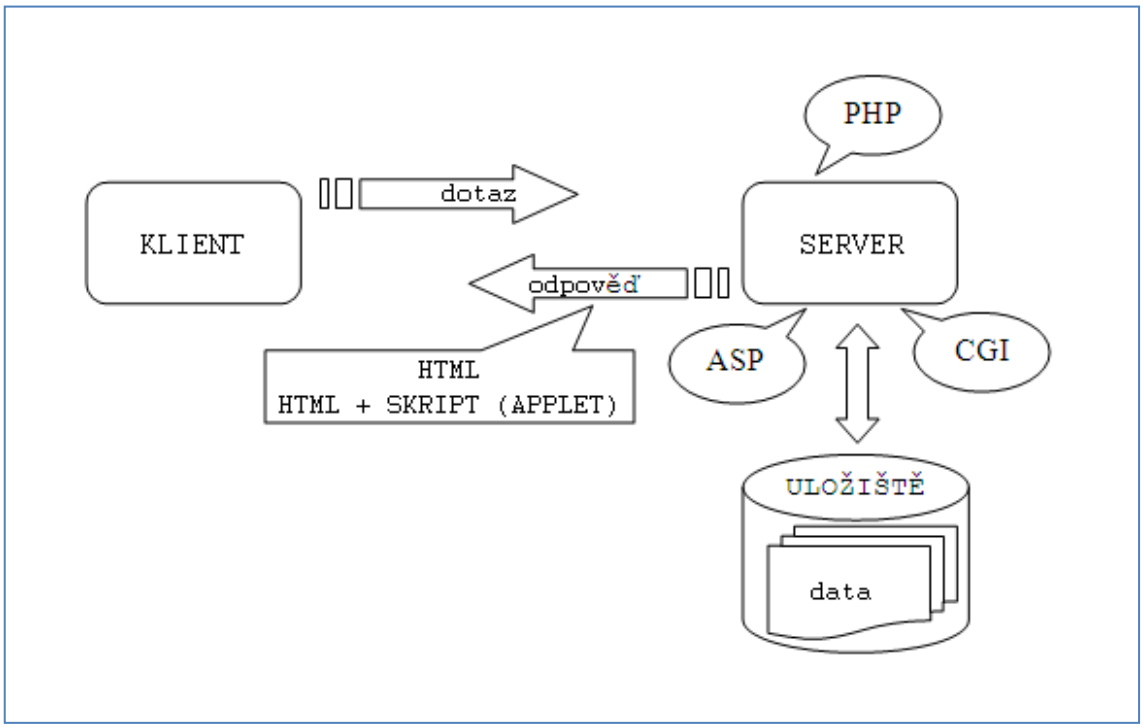
Webové stránky jsou hypertextové dokumenty přenášené v rámci World Wide Webu Internetem. World Wide Web, nebo jen zkráceně WWW je tedy jen jednou z mnoha služeb Internetu. Tato služba zajišťuje přenos mezi klientem a serverem. Zjednodušeně to funguje tak, že klient zašle žádost o dokument uložený na serveru a ten mu ho zašle. Průběh komunikace klienta se serverem znázorňuje obrázek *(obr. č. 1 – komunikace klient/server)* na následující straně. Pokud chce klient získat určitá data ze serveru, například v podobě zobrazitelné webové stránky na monitoru uživatele, zašle na server požadavek *(HTTP request)*. Tento požadavek má určitý formát dle specifikace použité verze HTTP protokolu. Součástí dotazu je jméno požadovaného dokumentu, kterému předchází umístění serveru v síti a lokace dokumentu přímo na serveru. Tato informace má podobu tak zvané URL adresy *(Uniform Resource Locator)*. Následuje příklad jejího rozboru:

http://skola.cernyjan.cz:80/diplomka/statistiky.html#server

tab. č. 1 – formát URL adresy

http://	protokol
skola.	doména III. úrovně
cernyjan.	doména II. úrovně
cz.	top doména (doména nejvyššího řádu)
:80	port
/diplomka/	adresář
statistiky.html	jméno dokumentu
#server	kotva (odkaz v rámci dokumentu)

Byl-li dotaz správně formulován, server na jeho základě započne patřičnou operaci, případně vrátí jedno z chybových hlášení.



obr. č. 1 – komunikace klient/server

Obsah dokumentů je psán značkovacím jazykem (X)HTML, v případě statických stránek jsou dokumenty uloženy v souborech na webu, jedná-li se o dynamicky generované webové stránky, tak jejich obsah vzniká až v době dotazu na server. O to se stará jeden ze skriptovacích jazyků, kterým může být například PHP. V současné době jsou do statických stránek vkládány klientské skripty JavaScript jazyka, které umožňují určitý druh dynamiky.

2.2.1 HTML

HyperText Markup Language, zkráceně jen HTML, je značkovací jazyk, kterým jsou psány hypertextové dokumenty v WWW. Jedná se o jednodušší aplikaci dříve vyvinutého, ale příliš komplexního jazyka SGML. Poprvé byl jazyk nasazen při vzniku World Wide Webu v roce 1990. Od té doby je verzí několik: od *HTML 2.0* přes *HTML 4.01* až *HTML 5* – nejnovější verze HTML jazyka, která se v současné době (r. 2012) s přibývající podporou v internetových prohlížečích stává čím dál více aktuálním tématem. Obsahuje nové tagy pro prvky stránky, jako je „hlavička“ a „patička“, nebo tagy pro vložení a následné přehrávání audia a videa bez instalace

dodatečných a nestabilních „plug-inů“ (zásuvných modulů – doplňkový software aplikace, který rozšiřuje její funkčnost).

2.2.2 XHTML

Jedná se o reformulaci HTML jako aplikace XML[2]. XML je obecný značkovací jazyk vycházející ze staršího jazyka SGML, který byl již zmíněn v souvislosti s HTML. „XML **nic neříká o sémantice**, významy prvků musí znát aplikace – program X podporuje XML, neznamená program X rozumí jakémukoli XML dokumentu.“[3]

Zkratka XHTML je tvořena z anglických slov ***ex**tensible **h**ypertext **m**arkup **l**anguage*. Překlad do českého jazyka zní – „rozšiřitelný hypertextový značkovací jazyk“, i když ve skutečnosti, XHTML oproti HMTL, přináší zúžení výběru dříve používaných značek pro tvorbu internetových stránek a nastolení vyšší restrikce. XHTML byl nástupcem zastaralejšího a benevolentnějšího značkovacího jazyka HTML pro tvorbu hypertextových dokumentů v prostředí World Wide Webu. Nyní už tomu tak není, jelikož je ve vývoji konsorciem W3C (World Wide Web Consortium) nová verze jazyka HTML, a to již s číselným označením 5, která by měla být standardizována v roce 2014.

2.2.3 PHP

Skriptovací programovací jazyk, kterého je využíváno při tvorbě dynamických WWW stránek a webových aplikací. Skripty se interpretují na straně serveru. V roce vzniku PHP tato zkratka znamenala ***P**ersonal **H**ome **P**age*, nyní se již jedná o rekurzivní zkratku odvozenou z anglického ***H**ypertext **P**reprocessor*.

Aby mohlo PHP fungovat, je potřeba k „běhu“ interpretu dvou věcí, webového serveru a PHP modulu. PHP je uvolněno pod licencí PHP License.[4]

Skripty se vkládají mezi značky ***<?php*** – uvozovací tag a ***?>*** – ukončovací značka. Tak server pozná, že kód v této oblasti nemá společně s HTML odesílat klientovi, ale že je nejprve tento fragment kódu třeba vykonat a na žádost odpovědět až tím, co skript „vrátí“.

2.2.4 ASP.NET

ASP.NET je nástupce technologie ASP. „ASP (Microsoft Active Server Pages) je skriptovací prostředí pro servery, které můžete využít pro tvorbu a provozování dynamických interaktivních webových aplikací. S využitím ASP, můžete kombinovat HTML stránky, skripty, a COM komponenty a vytvářet tak interaktivní webové stránky nebo výkonné webové aplikace, které se snadno vyvíjejí i upravují.“[5]

Oproti původní technologii, která podporovala tvorbu webových stránek pouze ve VBScript, je možné tvořit v jakémkoliv jazyku CLR *(Common Language Runtime)* kompatibilním – Visual Basic.NET, C# a další .NET mutace jazyků. Pro správnou funkčnost je vyžadována přítomnost .NET Frameworku a nejlépe Microsoft-IIS server.

2.2.5 JavaScript

Interpretovaný skriptovací jazyk, který je vykonáván až na straně klienta, v prohlížeči webových stránek. JavaScript není Java. Jedná se pouze o podobnost názvu z marketingového důvodu v jeho počátcích vzniku, i když v některých rysech má podobnou syntaxi. JavaScript je multiplatformním jazykem – spustitelný zdrojový kód pod více jak jednou konkrétní platformou.

Kód je přímo vkládán do HTML stránky. Aby prohlížeč poznal, odkud kam má JavaScript vykonávat, tak je skript uvozen HTML tagem ***<script>*** a zakončen jeho párovým uzavíracím elementem ***</script>***. V tagu musel být přítomen povinný atribut **type**, který nabývá hodnoty *text/javascript*, to s nástupem HTML ve verzi 5 vymizí, protože JavaScript se stává defaultním skriptovacím jazykem.

Oproti klasickým výhodám programovacího jazyku přináší tento objektově orientovaný jazyk silný nástroj, kterým je navigace v DOM *(Document Object Model – objektový model dokumentu, objektově orientovaná reprezentace HTML dokumentu)*, získávání a úprava obsahu elementů stránky. „DOM objekty jsou obvykle zobrazeny na monitoru, například okna, formuláře, tlačítka ap. Když chceme objekt použít, musíme použít jeho jméno. Jméno DOM objektu obvykle musí obsahovat předpony nadřazených objektů podle hierarchie objektů.“[6]

Pro demonstraci následuje ukázka změny atributu HTML elementu. Jedná se o změnu zdroje obrázku:

```
<img id="obrazek" src="portrait.jpg">


<script>
document.getElementById("obrazek").src="landscape.jpg";
</script>
```

V souvislosti s JavaScriptem se objevuje spojení JavaScript framework. Jedná se o ucelený soubor JavaScript funkcí a procedur, které mají uživateli sloužit k urychlení jeho práce tím, že přináší nástroje pro běžně užívané rutiny. Rozsáhlejší frameworky obsahují i nejrůznější komponenty do stránek, jako jsou kalendáře, automaticky se validující formuláře a podobně. JavaScript frameworků je k dispozici nepřeberné množství, některé z nich jsou placené, ale velké množství je poskytováno vývojářům bezplatně. Autoři dvou článků (první[7] a druhý[8]), kteří vybírali vždy několik podle nich nejpoužívanějších JavaScript frameworků, se shodli nad těmito frameworky:

- **jQuery** – je to malá a rychlá JavaScript knihovna, pro manipulaci s DOM a CSS, její součástí jsou animace a efekty spojené s událostmi prvků webové stránky, to vše je distribuováno v jednom souboru na stránkách *http://jquery.com/download/*, kde si vývojář stáhne verzi, kterou chce používat a tento soubor posléze nahraje na server spolu s webovou stránkou, případně lze použít dotazů na externí Google Hosted Libraries, pomocí AJAXu *(AJAX – **a**synchronous **J**avaScript **a**nd **X**ML)*

- **Prototype** – o několik desítek kilobytů méně zabírající JavaScript framework s obdobnou funkcionalitou jako jQuery, který neobsahuje nástroje pro tvorbu efektů, stejně jako předešlý framework, je poskytován formou jednoho souboru ze stránek *http://prototypejs.org/download/* nebo formou Google Hosted Libraries

- **MooTools** – nejmenší, avšak plnohodnotný nástroj pro práci s DOM, stejně jako předešlé frameworky je zahrnut v Google Hosted Libraries, případně ho lze stáhnout na *http://mootools.net/download*

Časové rozpětí mezi uveřejněním prvního (r. 2008) a druhého textu článku (r. 2012) je čtyři roky. A přesto výše zmíněné frameworky vystupují i ve článku druhém, který pochází ze stejného roku jako tato práce. Jedná se o stavební kámen, o který se bude opírat modul pro zjišťování používaných JavaScript frameworků výsledné aplikace této diplomové práce. V závěru práce se dozvíte, zda ten nejvíce využívaný framework ze vzorku testovaných webů je opravdu jedním z nich, nebo ho úplně nějaký jiný převyšuje.

2.3 Python

Programovací jazyk, u jehož vniku stál Guido van Rossum. „Jméno dostal podle pořadu BBC Monty Python's Flying Circus.“[9] Jedná se o ryze objektový jazyk, který je multiplatformní. Skripty v něm napsané lze spouštět nejen na desktopu, kde je přítomen Python interpreter, ale i na mobilních zařízeních, které „běží“ na Windows Mobile, Symbian a nově Android. V základu obsahuje mnoho užitečných modulů, například modul pro práci s HTTP nebo přístupu ke službám operačního systému. O další moduly lze Python rozšířit pomocí tzv. PyPI *(Python package index - https://pypi.python.org/pypi)*, jedná se o repozitář, kam lze uveřejňovat vlastní Python moduly, které si pak může každý stáhnout a naimportovat do svého projektu. Bližší informace o jazyku včetně rozsáhlé dokumentace doplněné o praktické ukázky na *http://python.org/doc/*.

2.4 Relační vs. Dokumentově orientovaná databáze

Databáze je kolekcí dat, která slouží k popisu reálného světa (například aktivity určité organizace – evidence vědecké knihovny a podobně). Jednotlivé databáze jsou uloženy na paměťovém médiu a operace s nimi jsou zajišťovány pomocí softwaru k tomu určenému, kterým je SŘDB (systém řízení báze dat – v českých odborných textech, případně DBMS z anglického database management system). Souhrnné označení pro soubor dat se softwarovými prostředky, které je spravují, se označuje DBS (databázový systém). Pokud se dále v textu vyskytne pojem databáze bez bližšího určení, bude se vždy jednat o úhrnné označení kolekce dat a softwarové prostředky, které je spravují.

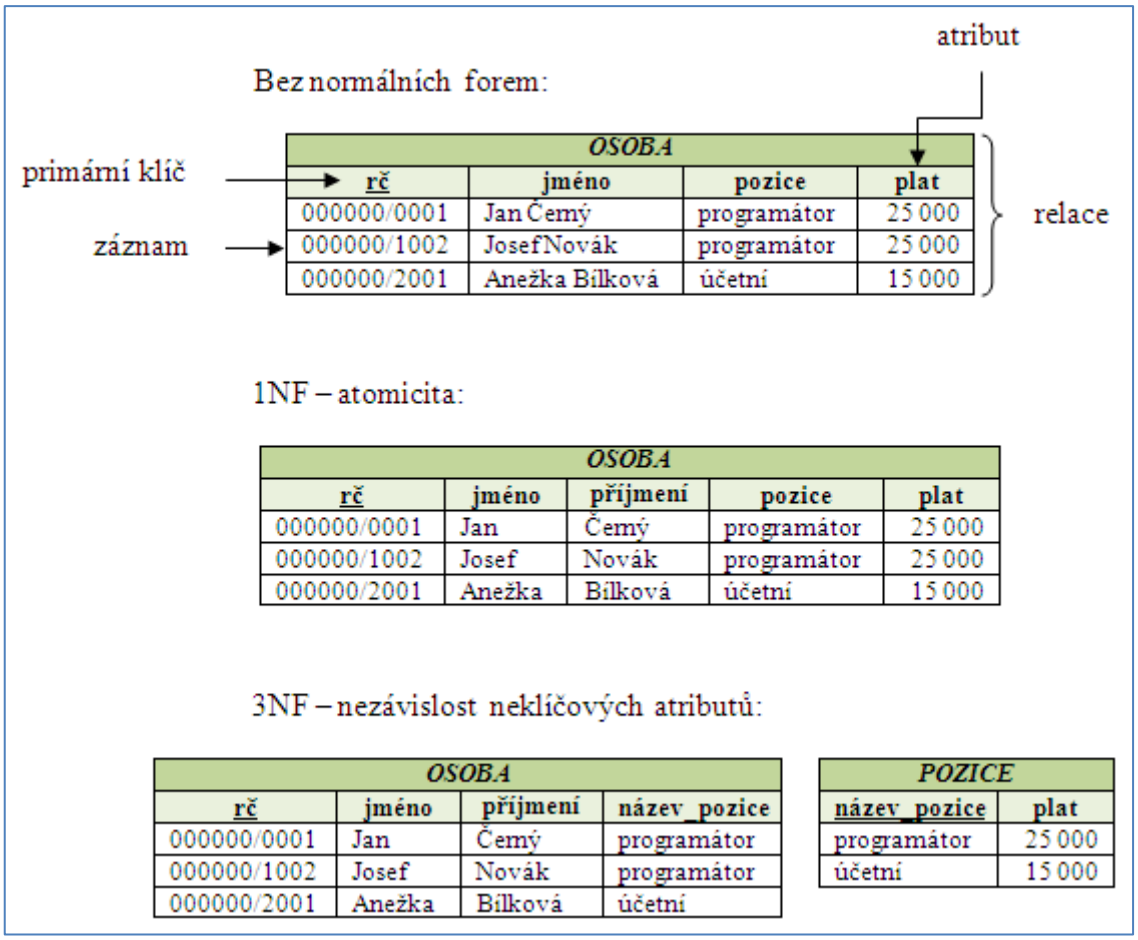
Aby mohl DBS správně fungovat a plnit svůj účel, musí mu být určeno, co bude ukládat. K tomu slouží datové modelování. „Při datovém modelování na základě znalostí o realitě navrhujeme struktury dat, do kterých se budou o této realitě ukládat záznamy. Je to činnost, která vyžaduje důkladnou analýzu skutečnosti, a schopnost převést získané představy do databázových struktur. V souhrnu se této činnosti říká návrh databáze.“[10]

Nejvíce používanou a početně rozsáhlou skupinou jsou DBS založené na relačním modelu dat, tzv. **Relační databáze**. Jejich počátek sahá až do roku 1970, kdy E. F. Codd představil svou vizi, postavenou na pevném matematickém základě, kde se na data hledí jako na matematickou relaci. V těchto databázích jsou základním stavebním kamenem tabulky (entity) a jejich atributy (názvy sloupců), nad kterými je pracováno pomocí prostředků relační algebry. Když je databáze navrhována, je třeba se vyvarovat některých chyb, jako je například redundance dat, která by způsobila nekonzistentnost dat. K tomu slouží normální formy. Není to podmínkou, ale většinou je snaha o třetí normální formu *(značeno 3NF),* která následuje po 1NF a 2NF. Databáze, která splňuje podmínky třetí normální formy, se vyznačuje:

- všechny hodnoty atributů jsou atomické – nelze je dále dělit beze ztráty informace (přejímá z 1NF)

- každý atribut, který není klíčem, musí být závislý zcela na primárním klíči (vychází z 2NF)
- všechny atributy, které nejsou klíčem, musí být mezi sebou nezávislé

Na následující straně je na obrázku *(obr. č. 2 – operace s normálními formami)* demonstrováno na konkrétním příkladu jak dostat databázi do třetí normální formy.



obr. č. 2 – operace s normálními formami

U relačního modelu dat musí být předem navrhnuta struktura dat, které budou poté do databáze ukládány. Musí být určena struktura tabulek a integritní omezení. Pokud se do budoucna vyskytne nějaká změna, musí dojít k úpravě tabulky, a to může mít neblahý vliv na celý původní návrh databáze. Této vlastnosti se snaží vyhnout novější datové modely, jedním z nich je **dokumentově orientovaná databáze.** Tato databáze je kolekcí dokumentů. Každý dokument se skládá z jedinečného identifikátoru a množiny dvojic klíč-hodnota, které lze dle potřeby dále vnořovat. Dokumenty jsou uchovávány zpravidla ve formátu JSON *(JavaScript Object Notation)* – „textový, na jazyce zcela nezávislý formát, využívající však konvence dobře známé programátorům jazyků rodiny C (C, C++, C#, Java, JavaScript, Perl, Python a dalších)“[11], či podobě XML.

Následuje stručná rešerše dokumentově orientovaných databází. Tento druh databází je oproti relačním značně mladý a není jich zatím tolik jako těch relačních. Relační databáze jsou vžité a ani zde na univerzitě se jiný datový model nevyučuje,

a proto stál před autorem této diplomové práce nelehký úkol, seznámit se s nástroji pro správu dokumentově orientovaných databází a na jejím základě zvolit tu nejvhodnější pro vlastní potřeby aplikace. Vedoucí práce upozornil na existenci několik dokumentově orientovaných databází, které by mohly být použity. Byly to CouchDB, ElasticSearch a MongoDB. Druhá jmenovaná je nejmladší a z toho důvodu byla z přehledu vyřazena, je možné, že se časem stane jednou z těch lepších, ale pro potřeby diplomové práce bylo třeba volit stabilní databázi s ne příliš častým vydáváním nových verzí.

Před samotnou rešerší dokumentově orientovaných databází – CouchDB a MongoDB, však ještě malá odbočka v podobě zástupce z řad relačních databází, kterým je MySQL databáze.

2.4.1 MySQL

Nejrozšířenějším zástupcem relačních databází je multiplatformní MySQL databáze, se kterou uživatel komunikuje za pomoci SQL příkazů. Na poli freehostingu *(bezplatné umístění webových stránek na cizím serveru s nízkou technickou podporou a negarantovanými službami)* se uživatel nesetká prakticky s jinou databází. MySQL pracuje na bázi komunikace klient/server. Nejzákladnější a nejdůležitější příkazy pro práci s daty v MySQL jsou:

- **SELECT** – vybírá jeden nebo více záznamů z tabulky v databázi
- **INSERT** – vkládá jeden nebo více řádků s hodnotami do tabulky databáze
- **UPDATE** – slouží k aktualizaci stávajících dat v tabulce databáze
- **DELETE** – maže jeden nebo více řádků s hodnotami v tabulce databáze
- **DROP TABLE** – slouží pro kompletní odstranění tabulky z databáze

2.4.2 Apache CouchDB

Apache CouchDB, zkráceně jen CouchDB je dokumentově orientovaný open-source multiplatformní DBS šířený pod licencí Apache 2.0 *(blíže k licenci 2.0[12])*. Jádro databáze je naprogramováno v programovacím jazyku Erlang. „Erlang je funkcionální programovací jazyk vyvinutý firmou Ericsson v roce 1987.“[13] CouchDB si jako data v databázi uchovává soubor JSON dokumentů, které jsou tzv. schema-free *(nesdílejí společné schéma, jako tomu je v relačních databázích)*. Dotazování funguje formou Map/Reduce – vytvářejí se tzv. pohledy *(views)*, trvalé i dočasné, jedná se o vytvoření dalšího dokumentu, který se bude skládat z požadovaných dat ze všech vhodných dokumentů.

Vlastnosti:

- ***škálovatelnost pomocí HDD***
  - škálovatelnost – rozšíření systému o hardwarové prostředky, aby bylo možné dosáhnout větší propustnosti, zde pomocí HDD na rozdíl od CPU, které mají vyšší pořizovací náklady

- ***RESTful HTTP API***
  - REST – Representational State Transfer
  - pro komunikaci s databázovým serverem CouchDB je používáno rozhraní komunikující přes HTTP protokol, tak jak je známo pro práci se zdroji ve World Wide Webu
  - metodami GET, POST, PUT a DELETE s příslušnou URI lze dokumenty z databáze získávat, ukládat, měnit, případně spouštět různé funkce jako je komprese objemu dat na serveru a podobně
  - jednoduchá manipulace přes příkazovou řádku, komunikace s databázovým strojem v libovolném programovacím jazyku formou HTTP dotazů, grafické rozhraní pro správu jménem Futon přístupné z webového prohlížeče, které využívá RESTful a je součástí základní instalace CouchDB

- ***schema-free*** – dokumenty v rámci jedné databáze nesdílejí společné schéma, jako tomu je v relačních databázích

- ***append-only*** – každá změna v dokumentu je provedena založením nového dokumentu s původními daty, a touto změnou

Domovská stránka: *http://couchdb.apache.org/*

2.4.3 MongoDB

MongoDB je další dokumentově orientovaný open-source multiplatformní DBS. Jádro databáze je naprogramováno v programovacím jazyku C++. V MongoDB se každá databáze skládá z kolekcí *(obdoba tabulek v relačních databázích)*, do té se poté ukládají jednotlivé dokumenty. Z venku se chovají jako JSON dokumenty v CouchDB, ale uvnitř databáze se jedná o BSON *(binární serializace JSONu, BSON je navržen tak, aby byl lehce přenositelný a efektivní*[14]*)*. Obdobně jako u CouchDB lze využít Map/Reduce, ale co MongoDB přidává navíc je funkce *find()* – obdoba klausule WHERE z relačních databází. Poté je možné i v této databázi dohledat hodnotu podle klíče. Tento přístup není v dokumentově orientované databázi typický.

Vlastnosti:

- ***škálovatelnost pomocí RAM***

- ***mongod*** – oproti CouchDB, která má RESTful HTTP API, případně pro začínající uživatele snadné a přehledné grafické rozhraní Futon, databáze MongoDB ve své instalaci obsahuje javaScriptovou konzoli mongod, nebo je třeba stáhnout knihovnu pro příslušný programovací jazyk

- ***schema-free***

Domovská stránka: *http://www.mongodb.org/*

2.5 Automatický sběr dat

Pro automatický sběr dat v rámci WWW se používá specializovaný nástroj, který se nazývá internetový robot. Je možné se setkat se zkráceným pojmenováním internetový bot, případně je tento nástroj nazýván spider, či crawler. Jedná se o počítačový program navržený ke stažení stránek z WWW pro pozdější zpracování. Program nebo skript metodicky prochází World Wide Web automatizovaným způsobem.[15]

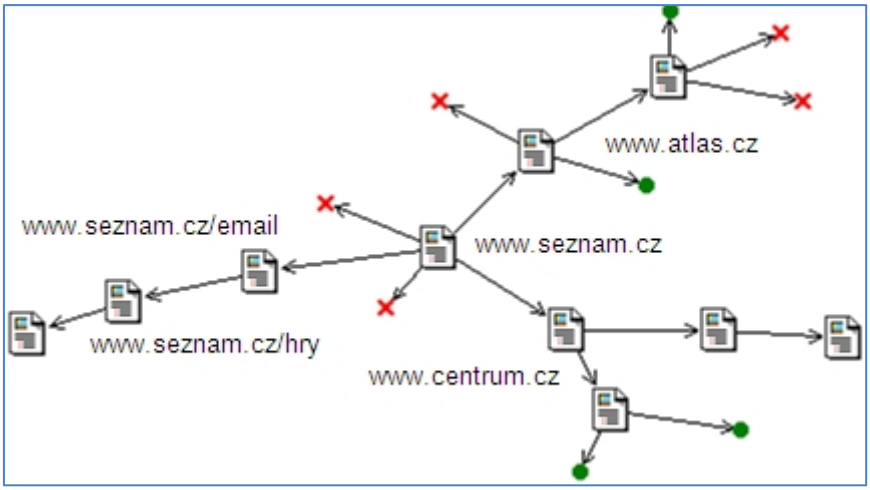
Robot pracuje tak, že na začátku dostane výchozí URL adresu, nejlépe takovou, která je rozcestníkem, například katalog internetové služby *Seznam.cz* nebo podobné. Crawler každou navštívenou stránku stáhne a spolu s její URL adresou ji uloží, aby ji neprocházel znovu.

Algoritmus internetového robota:

```
Inicializace fronty (Q) URL adres

Until Q neprázdná, nebo nedosazen časový/početní limit:
   Pop URL (L) z Q
   If L není HTML stránka (pdf, jpg, ps, …)
      konec otáčky
   If L již navštívena, pokračování otáčky (další L)
   Stažení stránky (P) pro L
   If nelze stáhnout P (například 404 error, robot.txt, …)
      konec otáčky, alternativní větev
   Indexace P (uložení)
   Parsování P pro získání nových URL (N)
   Zařazení N na konec Q
```

Průchod internetového robota World Wide Webem muže být jak do šířky, bot prochází pouze doménové stránky, které na sebe odkazují (jedná se o stránky na stejné úrovni – *http://www.seznam.cz*, *http://www.atlas.cz*, …), tak do hloubky, kdy v každé navštívené doméně prochází kompletní adresářovou strukturu na serveru a zaznamenává na něm každou stránku (*http://www.seznam.cz/email*, *http://www.seznam.cz/hry*, *http://www.lide.seznam.cz*, …).



obr. č. 3 – cesta internetového robota

Chování internetového robota při průchodu World Wide Webem je na základě jedné z možných strategií. V praxi se jedná o jejich kombinaci.

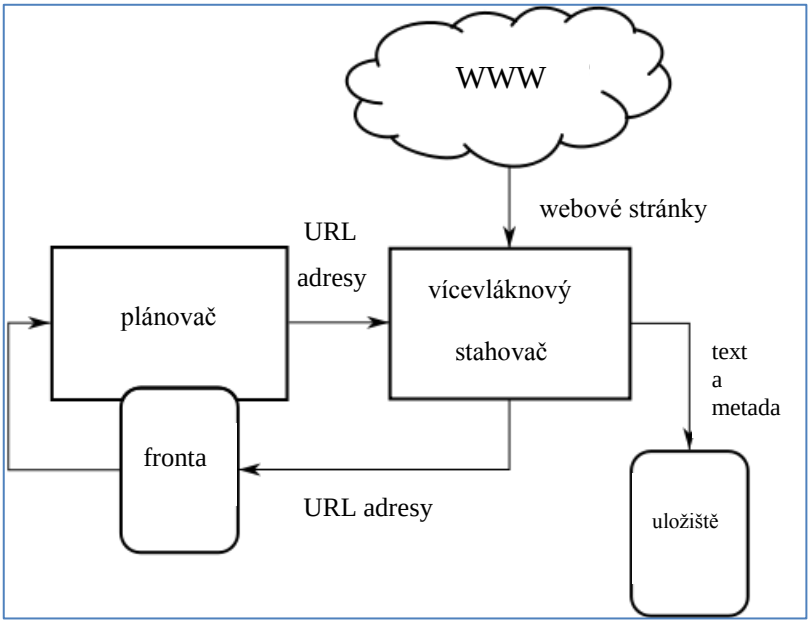
- Na základě výběru (***Selection policy***) – tato strategie uvádí, které WWW stránky budou robotem staženy. Robot obsáhne jen malý zlomek skutečného objemu World Wide Webu, proto je důležité zaměřit se skutečně jen na ty z určitého důvodu zajímavé.

- Opakovaná návštěva (***Re-visit policy***) – WWW je velice dynamická služba, kde se obsahy hypertextových dokumentů často mění, proto je třeba stránky opakovaně po určité době navštěvovat a kontrolovat na nich změny.

- Zdvořilostní taktika (***Politeness policy***) – udává, jak se vyhnout přetížení webových stránek. Roboti mohou mít zničující dopad na celkovou výkonnost stránek. Částečným řešením tohoto problému je protokol pro zakázání přístupu robotům. Jedná se o textový soubor *robots.txt* umístěný v kořenovém adresáři webových stránek. V tomto souboru je nadefinováno, kteří roboti mají povolený přístup a kam mohou. Zákaz zaindexování konkrétní stránky lze určit tagem *<meta name="robots" content="noindex,nofollow" />* v HTML dokumentu.

- Paralelizační řízení (***Parallelization policy***) – uvádí, jak koordinovat distribuované internetové roboty. Jedná se o případ, kdy „běží“ paralelně více robotů. Hlavním cílem je maximalizovat rychlost stahovaní stránek a minimalizovat režii z paralelizmu. Dále je snaha, aby se předešlo opakovanému stažení stejné stránky.

Na obrázku *(obr. č. 4 – schéma internetového robota)* je zachyceno obecné schéma internetového robota. Komponenta fronta obsahuje nenavštívené URL adresy, plánovač rozhoduje, které URL adresy poskytne stahovači. Výběr je na základě krátkodobého, nebo dlouhodobého plánování *(short-term scheduling a long-term scheduling algoritmus).*[16] Určuje frontu webových domén, nebo stránek, které na nich sídlí. Vícevláknový stahovač zajišťuje vlastní stahování obsahu WWW stránek z World Wide Webu. Uložiště slouží pro uchování hypertextových dokumentů.



obr. č. 4 – schéma internetového robota

2.6 Data mining

Data mining, slovní spojení, které je do češtiny překládáno jako dolování dat, případně vytěžování dat, je procesem získávání nových, pro nás do té doby neznámých, informací z našich dat. „Data mining je velmi propracovaná metoda, která pomocí matematických funkcí analyzuje velké objemy dat a hledá v nich skryté souvislosti.

Nejčastěji se používá v marketingu a její výsledky slouží velkým pojišťovnám, supermarketům nebo mobilním operátorům.“[17] Celý proces bývá rozložen do 3 částí:

- **Pre-processing** – předběžné zpracování, ve kterém dochází u dat, nad kterými má být prováděn data mining, k odstranění nepodstatných informací, které obsahují ve své původní podobě

- **Data mining** – vlastní proces vytěžování dat na základě použitého algoritmu

- **Post-processing** – dodatečné zpracování získaných dat z předchozí fáze, například jejich grafová reprezentace a podobně

V této práci bude probíhat vytěžování dat nad dokumentově orientovanou databází, která bude obsahovat záznamy v podobě uložených hypertextových dokumentů přístupných z domén II. řádu. Jednotlivé konkrétní fáze data miningu pro tuto diplomovou práci se nacházejí v kapitole *Praktická část*.

Vytěžování dat z prostředí WWW se nazývá *Web mining*. Web mining si klade za cíl zjistit užitečné informace nebo znalosti z hypertextové struktury, obsahu stránky a údajů o používání.[18]

2.7 Existující realizace

V této podkapitole je krátká rešerše nástrojů pro automatický sběr dat z WWW a přehled stávajících projektů, které se více či méně zabývají podobnou problematikou jako tato diplomová práce. Přes veškerou snahu nebyl nalezen jediný český projekt, proto rešerše obsahuje pouze světové projekty.

2.7.1 pyspider

První testovaný crawler byl zdrojový kód napsaný v programovacím jazyce Python a uvolněný pod licencí GPL. Tento jednoduchý skript se spouštěl z příkazové řádky s parametrem výchozí URL adresy. Nevýhodou aplikace *pyspider* bylo, že neprocházela další adresy ve World Wide Webu, ale pouze křížové odkazy v rámci výchozí lokace.

2.7.2 WebSPHINX

Dalším nástrojem byl *WebSPHINX* napsaný v prostředí jazyku Java. Tato aplikace poskytuje uživateli grafické rozhraní, přes které se ovládá. Pohyb „sběrače“ lze řídit ve 3 rovinách. První z nich je pouze aktuální adresa a všechny její podadresáře na serveru, druhou rovinou je pohyb v rámci celé domény a nejvyšší je celý WWW prostor.

Aplikace pracuje v několika módech. První z nich pouze graficky zobrazuje strukturu navštívených stránek, druhý mód pracuje jako „stahovač“ obsahu WWW stránek, které ukládá na disk. Jeden z dalších módů extrahuje data mezi konkrétními HTML značkami, pokud by šlo ovlivnit extrahování pomocí regulárních výrazů pro získání pouze určitých hypertextových odkazů, byla by aplikace silným nástrojem.

2.7.3 W3Techs

Prvním projektem, o kterém pojednává i článek na Z*droják.cz* v souvislosti s četností používání PHP je *W3Techs* – World Wide Web Technology Surveys. Jak je možné se dočíst na jejich webové stránce, poskytují informace o používaných technologiích ve World Wide Webu. Mezi dostupné informace patří používané CMS *(Content Management System – systém pro správu obsahu webových stránek)*, skriptovací jazyky na straně serveru, na straně klienta, jaké JavaScript frameworky jsou použity, značkovací jazyk, jazyková sada dokumentů, nejčetnější formát, ve kterém jsou uloženy obrázky, na jakém webovém serveru web „běží“ a ještě několik dalších informací.

Procentuální zastoupení použitých technologií je dostupné zdarma, avšak pro získání informací, která webová stránka konkrétně například používá jQuery již musí zákazník zaplatit.

Domácí stránka projektu: *http://w3techs.com/*

2.7.4 HTTP Archive

Dalším projektem je *HTTP Archive*, který jde trochu jiným směrem než předešlý projekt. Předkládá informace spojené s HTTP. Uživatel se převážně dozví, kolik procent z celkového množství analyzovaných webových stránek skončilo při dotazu chybovou odpovědí (4xx nebo 5xx), případně kolik vede na jiný zdroj (3xx),

dále pak průměrnou velikost stránky (samotné HTML, CSS nebo JavaScript kód) a jako předešlý projekt vede statistiku nejběžněji používaných formátů obrázků.

Oproti W3Techs je možné dohledat výsledky pro konkrétní webovou stránku, ale obrácenou formu – k chybě 4xx dohledat všechny domény, již webová aplikace nepodporuje. Funkcí navíc je i přehled nejvíce analyzovaných domén II. a II. řádu. Jejich počet je 10 000. Tohoto výpisu je možné dále využít pro vlastní potřeby, jako základu pro analýzu stránek.

Domovská stránka: *http://httparchive.org/*

2.7.5 BuiltWith

Posledním projektem je *BuiltWith*, který není zacílen jen na webové technologie, ale poskytuje široké spektrum analýz, jako je například zastoupení e-mailových nebo hostingových poskytovatelů, žebříček poskytovatelů platebních transakcí a mnoho dalšího. V rámci rešerše byly zkoumány služby, které poskytuje v souvislosti s tématem diplomové práce. Jedná se o procentuální zastoupení používaných webových serverů a JavaScript frameworky, i když u nich se klient nedozví v jaké míře je ten který použit, ale v jakém odvětví se používá, zda na stránkách zaměřených na cestování, business nebo prodej.

Většina bližších informací je na stránce zpoplatněna. Ze všech testovaných projektů se zdál tento jako nejméně přehledný, ale to může být do jisté míry způsobeno tím, že se snaží cílit na větší skupinu uživatelů a nabízí statistiky i z jiných oblastí, než bylo pro tuto práci třeba.

Domovská stránka: *http://builtwith.com/*

3 Praktická část

Kapitola pojednává o praktické stránce diplomové práce. Je v ní popsáno, jak aplikace pro automatické získávání dat z WWW vznikala. Od jejího návrhu až po konečnou implementaci v programovacím jazyku Python. Po té následuje stručné objasnění, jak s aplikací zacházet (kompletní softwarová dokumentace ke zdrojovým kódům je součástí příloh) a jakou formou jsou prezentovány výsledky výzkumu. V závěru kapitoly jsou uvedeny výsledky analýzy – poslední bod zadání diplomové práce.

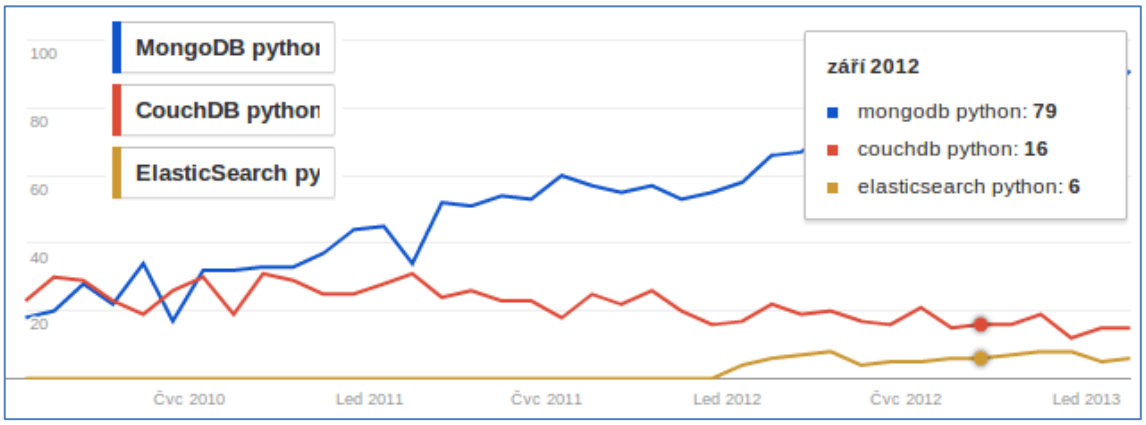
3.1 Před návrhem aplikace

Ze všeho nejdříve bylo třeba zjistit, zda už někdo podobný problém řešil. Pokud ano, existují-li někde na internetu výsledky obdobného výzkumu, případně přímo nástroje pro jeho opětovné provedení. Tímto se zabývá podkapitola *2.7 Existující realizace*, v teoretické části této technické zprávy. Z rešerše vyplynulo, že nástroje vhodné pro uskutečnění zadaných cílů diplomové práce neexistují a je na řešiteli, aby navrhl a implementoval vlastní řešení pro naplnění cílů diplomové práce.

Před samotným návrhem aplikace pro automatické získávání informací o World Wide Web stránkách bylo na řešiteli rozhodnout, jakou formu ukládání dat zvolit. Je zřejmé, že podstatnou část záznamů, se kterou bude aplikace pracovat, tvoří seznam URL adres. Zde je na výběr několik přístupů. Prvním z nich je, předat aplikaci výchozí adresu jako parametr a další získané při procházení WWW uchovávat v paměti. Z hlediska výkonnostního s ohledem na rychlost je tento přístup dobrý, ale z pohledu množství potřebné volné operační paměti je značně limitován – aplikace by měla zvládnout zanalyzovat nejméně 1 milion WWW stránek. Toto řešení přináší i jiný problém – přerušením výkonu aplikace způsobené výpadkem internetového připojení nebo napájení počítače. Druhou variantou je ukládat adresy do souboru. Co adresa, to jeden řádek. Oproti způsobu popsanému výše řeší problém s náhlým pádem aplikace a závislosti na dostupné operační paměti, v počátcích běhu aplikace. Pokud by se mělo v souboru vyhledávat a provádět s ním náročnější operace než jen ukládat nové záznamy, nahrával by se do paměti v podobě určité struktury a zvyšoval paměťové nároky aplikace. Dalším problémem je vyšší latence, která by se projevila při práci

se souborem – otevřít soubor, zapsat do něho a zavřít soubor. Výhodou je uchování dat i po ukončení aplikace.

Kompromisem mezi oběma přístupy k uchovávání dat je použití databázového uložiště. Z pohledu nahlížení na data jsou rozlišovány hlavní dva typy. Relační model dat a dokumentově orientovaný. Více o rozdílech mezi nimi v podkapitole *2.4 Relační vs. Dokumentově orientovaná databáze*. Pro potřeby aplikace bylo vhodnější zvolit dokumentově orientovanou databázi – souvislost mezi dokumentem v databázi a HTML dokumentem ve World Wide Webu. Typ databáze byl zvolen, ale konkrétní DBS musel řešitel diplomové práce teprve vybrat. V povědomí řešitele byly tři dokumentově orientované databáze. První z nich byla CouchDB, dále ElasticSearch a MongoDB.



obr. č. 5 – Google Trends pro spojení DATABÁZE-PYTHON

Databázový systém ElasticSearch, jak znázorňuje graf ze služby Google Trends – „Služba Trends od společnosti Google umožňuje zobrazit četnost vyhledávání určitého klíčového slova. Klíčových slov může být i více a vy je tak můžete vzájemně porovnat. Tato služba se vám hodí, pokud chcete zjistit zájem lidí.“[19], která byla použita pro hrubý přehled četnosti dotazů na spojení DATABÁZE-PYTHON, má skóre nejnižší. Na základě toho grafu a rešerše zbylých dvou dokumentově orientovaných databází byla volena CouchDB. Pro její konečnou volbu byla nejvíce rozhodující přítomnost grafického uživatelského rozhraní Futon a množství python knihoven pro její obsluhu z vlastní aplikace.

3.2 Návrh aplikace

Řešený problém byl rozdělen do dvou hlavních částí. První z nich je získání dat, které se skládá ze sběru URL adres a indexování stránek. Druhou částí je data mining – operace nad touto množinou dat a získání nových dat (informace o WWW stránkách). V jednotlivých podkapitolách bude vždy rozebrána jedna konkrétní část návrhu aplikace.

Jak z rešerše existujících projektů vyplynulo, podkapitola *2.7 Existující realizace*, obdobné realizace existují, ale obsahují některé nedostatky, případně se při analýze World Wide Web stránek zaměřují ve svém zkoumání na jinou problematiku. Z pohledu řešitele této diplomové práce je hlavním nedostatkem omezená dostupnost informací o konkrétních stránkách, případně rozšíření základních služeb po zaplacení jistého finančního obnosu. Těmto nedostatkům by se chtěl vyhnout a poskytnout vlastní přijatelnou alternativu.

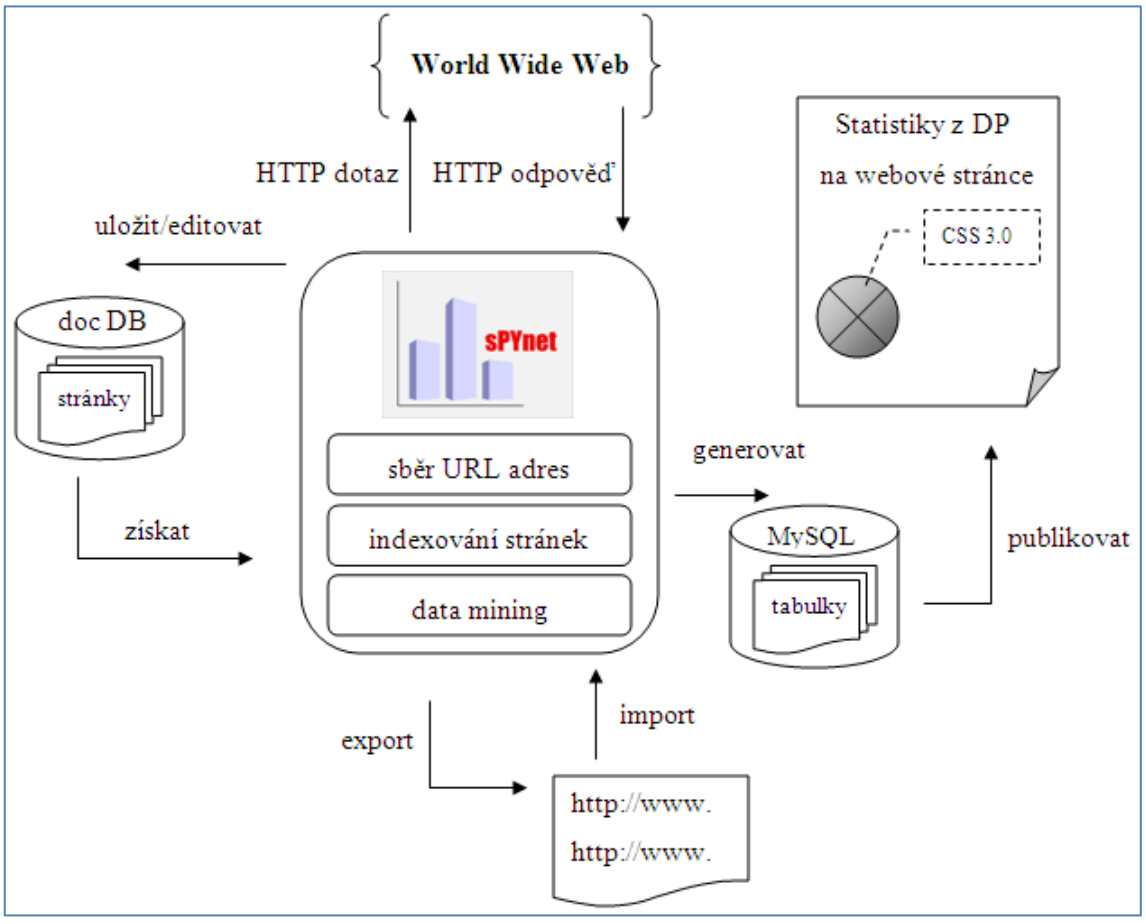
3.2.1 Získání dat

Data, bez nich nelze aplikací nic zjišťovat a vyhodnocovat. Získání dat, přednaplnění databáze daty, je pro tento druh aplikace, která provádí data mining, jednou z klíčových operací. Pokud jsou data již k dispozici, lze tuto operaci vynechat a pokračovat na jejich zpracování. O tento případ se jedná v době, kdy bylo minimálně první měření provedeno a nyní nad stejnými daty bude použita jiná sada metod pro zjištění nových informací – modularita aplikace.

Výchozími daty pro aplikaci budou URL adresy na jednotlivé WWW stránky ve World Wide Webu. Z těchto umístění budou poté stahovány a ukládány do databáze. Vedoucím práce byla stanovena omezující podmínka, že se bude pracovat se stránkami sídlícími pouze na doménách II. řádu. Jedná se například o URL adresu *http://www.seznam.cz*. Kde *www* není subdoménou – doménou III. řádu, ale výchozí veřejně dostupný adresář domény II. řádu. Mimo této podmínky si řešitel diplomové práce stanovil další dvě. První z nich vychází z podmínky předešlé. URL adresy, které aplikace akceptuje, budou pouze ve tvaru s *www*. Webový portál zmíněný v příkladu je dostupný jak v zápise s *www*, tak i bez *www*. Toto by mohlo vést k duplicitě nasbíraných dat, jelikož primárním klíčem v databázi bude URL adresa, která by se lišila, ale obsah stránky by byl totožný. Proto byla striktně zavedena tato podmínka, aby tento problém nenastal. Druhou podmínkou je, že pro analýzu WWW

stránky bude vždy použit tzv. index stránky – *http://www.domena.cz/**index.html***, **index.php** a obdobné. Tuto stránku server klientovi předává i v případě, že se na příslušný hypertextový dokument přímo nedotáže a vstupuje pouze na adresu domény. Toto pravidlo bylo zvoleno z několika důvodů. Prvním důvodem byly paměťové nároky na databázi. Druhým čas potřebný k zpracování tak rozsáhlých dat. Posledním důvodem, byla skutečnost, že každá další stránka sídlící na stejné doméně má shodnou verzi serveru, který odpovídá na dotazy. Jedním ze zkoumaných parametrů je právě informace, na kterém serveru stránka „běží“, množstvím dat by se informace nezpřesňovala. Vzhledem k tomu, že index stránka je vždy brána jako úvodní stránka, která klientovi poskytuje nejobsáhlejší informace, měla by pro její vznik být použita stejná technologie tvorby stránek jako na všechny ostatní, ze které je na ně možné se v rámci domény dostat, proto je tato stránka pro potřeby aplikace dostačující.

Pro získání rozsáhlé databáze stránek je nutné aplikaci poskytnou výchozí URL adresu, ze které akce sběru dat započne. Adresu je nutno volit tak, aby stránka, na kterou odkazuje, obsahovala velké množství externích odkazů v rámci WWW.



obr. č. 6 – schéma návrhu

Na obrázku *(obr. č. 6 – schéma návrhu)* je zachyceno komplexní schéma návrhu celé aplikace včetně všech rozhraní, vstupů a výstupů. Po spuštění začne aplikace postupně procházet World Wide Web z výchozí URL adresy. Pohyb mezi dalšími stránkami bude realizován na základě hypertextových odkazů, které jsou součástí každého HTML dokumentu, který aplikace od dotazovaných serverů obdrží.

3.2.2 Operace nad daty

Po úspěšném naplnění databáze daty – HTTP hlavičky a HTML kódy příslušných stránek, přejde aplikace do druhého módu, kterým je vlastní data mining. V této fázi postupně aplikace prochází všechny dokumenty databáze, nad kterými spouští jednotlivé metody pro získání informací, které uživatel zvolí při jejím spuštění. Maximální počet dostupných informací o každé WWW stránce je pět. Jsou to tyto informace: název serveru, na kterém World Wide Web stránka „běží“, značkovací jazyk a jeho verze, který byl použit pro tvorbu stránky a verze kaskádových stylů. Dalšími informacemi jsou, skriptovací technologie na straně serveru a na straně klienta – jeden z možných JavaScript frameworků, které aplikace rozpoznává.

3.3 Implementace návrhu

Tato podkapitola popisuje implementaci návrhu aplikace pomocí programovacího jazyka Python. Tento programovací jazyk byl použit na základě jednoho z bodů zadání diplomové práce.

3.3.1 Sběr URL adres

Sběr URL adres začíná z výchozí URL adresy, kterou je nutné aplikaci poskytnout. K tomu slouží jako vstupní parametr do aplikace cesta k textovému souboru, ve kterém je požadovaná adresa. Vhodnou adresou je taková URL, která obsahuje velké množství externích odkazů v rámci WWW. Například *www.seznam.cz* nebo *www.popurls.com*, které obsahují velké množství externích odkazů. Vzhledem k tomu, že není vždy snadné zvolit tu správnou výchozí URL adresu, je možné aplikaci v tomto textovém souboru předat více adres. Pro správný chod aplikace, je nutné soubor vytvořit způsobem, co řádek to jedna URL

adresa. Tohoto importu adres je možné využít i v případě, že uživatele zajímají konkrétní stránky, na kterých chce provést měření.

Ukázka spuštění aplikace, zvýrazněný parametr slouží pro dodání výchozí URL adresy, nebo import URL adres pro analýzu konkrétních WWW stránek:

```
python main.py –N db01 2300 urls.txt –S –H –C –ST –F -dg
```

Aplikace vstupní soubor kontroluje. Platí pro něj omezení zmíněná výše, je třeba dodržet správný tvar URL adres v něm uložených.

Aby mohla aplikace data ukládat a následně s nimi pracovat, musí být v počítači, ze kterého je aplikace spouštěna přítomen databázový systém, případně před spuštěním samotné aplikace upravit cestu k databázovému serveru na jeho externí lokaci. Aplikace pracuje s dokumentově orientovanou databází, konkrétně s Apache CouchDB. Ukládané dokumenty jsou ve formě JSONu. Každé tělo dokumentu se skládá z dvojice klíč-hodnota. Hodnota příslušející ke klíči *_id* (defaultní klíč generovaný při založení nového dokumentu) je vždy příslušná URL adresa. Hodnota klíče *_rev* je opět generována automaticky databázovým strojem a váže se k jednotlivým revizím dokumentu – při každé změně dokumentu je hodnota měněna.

```
{
    „_id“: „http://www.seznam.cz“,
    „_rev“: „1-6385cb629f570f1f1ef251caf11f7370“,
    „code“: „null“
}
```

Ukázka demonstruje, jak vypadá aplikací nově vytvořený dokument v databázi. Vždy když aplikace při sběru dat narazí na URL adresu, kterou nemá v databázi, vytvoří pro příslušnou World Wide Web stránku dokument, kde do klíče *_id* přiřadí hodnotu této URL adresy a klíči *code* hodnotu **null**.

Sběr URL adres probíhá formou, že na každé aktuálně „navštívené“ stránce, jejíž kompletní obsah aplikace uloží do databáze, aplikace vyhledá všechny URL adresy podle daného předpisu ve formě regulárního výrazu, který je součástí funkce *extract_urls*:

```
...
regex = re.compile(r"http://(?:www\.)(?:[^-]{1}[a-z0-9-
]+\.){1,3}[a-z]{2,6}")
reg = regex.findall(string)
return list(set(reg))
...
```

Takto získanou množinu převede na neuspořádanou n-tici URL adres, ze které následně vytvoří seznam obsahující pouze URL adresy domén II. řádu, viz omezující podmínky. K tvorbě tohoto seznamu slouží funkce *get_top_domains* využívající balíku *urlparse*, který je součástí jazyka Python. Podrobnější informace o této funkci v softwarové dokumentaci ke zdrojovému kódu aplikace, která je součástí příloh této technické zprávy. Takto vytvořený seznam aplikace vždy projde a pro každý prvek vygeneruje HTTP dotaz metodou HEAD. Na základě odpovědi od serveru (pokud byl kladně vyřízen) URL adresu uloží. Tato kontrolní funkcionalita byla přidána až později, jelikož byla databáze plněna URL adresami, které nebyly dostupné. Pro zvýšení výkonu aplikace se při uložení nového dokumentu netestuje, zda databáze dokument neobsahuje a na základě tohoto zjištění by se dokument uložil, ale rovnou je dokument ukládán. Databáze původní dokument nepřepíše, jen vrátí informaci o tom, že dokument se stejným *_id* – URL adresou, již v databázi je, kterou aplikace odchytí a pokračuje dále.

Tato celá činnost se opakovaně provádí, dokud databáze neobsahuje takové množství dokumentů, jaké bylo aplikaci předáno pomocí parametru při jejím spuštění:

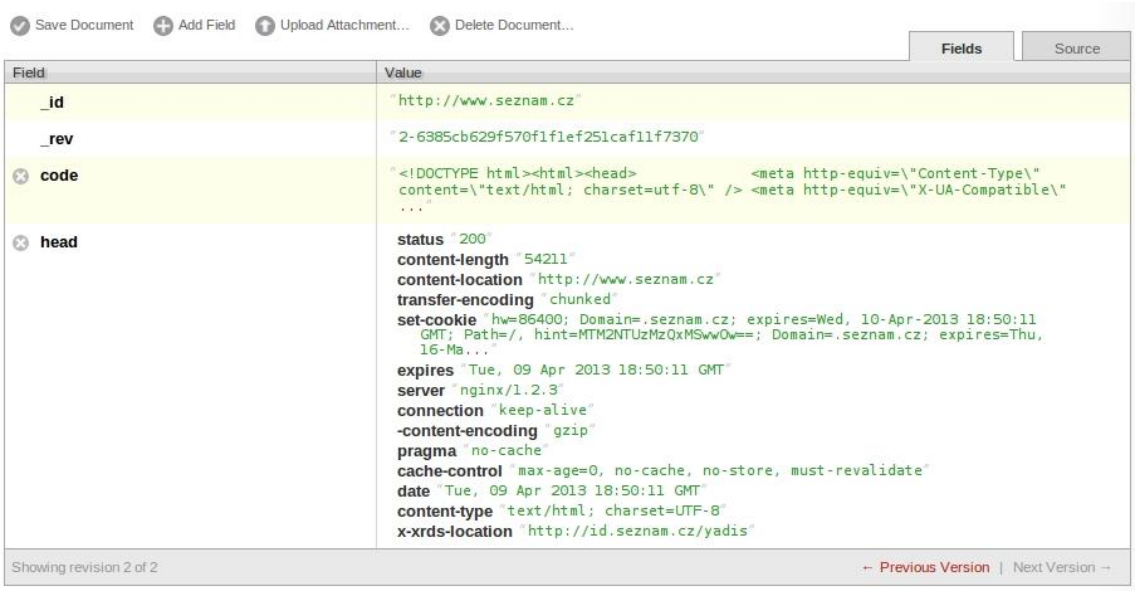
```
python main.py –N db01 2300 urls.txt –S –H –C –ST –F -dg
```

Při každé smyčce vyhledávání nových URL adres se jako další adresa bere klíč dokumentu, který obsahuje hodnotu **null** v klíči *code*. Po získání nových adres je jeho hodnota změněna na HTML kód aktuální stránky, který aplikace na dotaz od serveru obdržela.

3.3.2 Indexování stránek

Po naplnění databáze příslušným množstvím dokumentů, se spouští druhá fáze získávání dat – indexování stránek. Jedná se o činnost, kdy jsou procházeny všechny dokumenty databáze, které obsahují hodnotu **null** v klíči *code*. Při této operaci se již nevyhledávají nové URL adresy, ale pouze získávají HTML kódy a HTTP

hlavičky, nad kterými v další fázi budou probíhat metody pro získání informací o stránce. Po této fázi je databáze plně připravena pro data mining. Následující obrázek znázorňuje, jak vypadá zaindexovaná stránka v databázi:



obr. č. 7 – dokument zobrazený ve Futonu

V první verzi aplikace bylo získávání dat řešeno formou sekvenčního procházení databáze, dokument po dokumentu, o které se staral jeden proces. Od tohoto řešení bylo brzy upuštěno pro jeho nízkou efektivitu. Dalším způsobem jak zrychlit získávání dat bylo přidat druhý proces, který procházel databázi z jejího opačného konce než první proces. Tyto procesy „běžely“ současně. Tím se sice rychlost zdvojnásobila, ale i přesto by trval sběr URL adres a jejich následná indexace při počtu jednoho milionu příliš dlouho. Implementovat aplikaci jako multiprocesovou (paralelní zpracování dat formou více procesů) s sebou přinášelo problémy v podobě synchronizace a větších paměťových nároků. Kompletním vyřešením problému s rychlostí a synchronizací bylo navrhnout aplikaci od základu jako multithreadovou (vícevláknovou). V původním návrhu byly velké časové prodlevy způsobené tím, že v každém cyklu se postupuje přes: dotaz na server → odpověď serveru (HTML dokument) → uložení dokumentu do databáze → zpracování HTML kódu (získání nových URL adres) → test každé URL adresy a její následné uložení do databáze. Pokud není vždy předchozí krok operace dokončen, čeká se s dalším. U více vláknového přístupu je spuštěno více vláken s touto operací pro různé URL adresy. Pokud je jedno vlákno zaměstnáno jedním krokem operace, ostatní vlákna

pracují a není třeba čekat na dokončení kroku. Protože jednotlivé kroky nemají stejnou dobu trvání, na některé stránce je hypertextových odkazů více, na jiné méně a další nemusí obsahovat žádné, bylo přínosné přistoupit k paralelismu. Problém se souběhem, protože vlákna pracují se sdílenou pamětí, je řešen pomocí zámků, které Python poskytuje.

3.3.3 Vytěžování informací

Naplněnou databázi daty – HTTP hlavičky a HTML kódy příslušných stránek, aplikace postupně prochází a nad každým dokumentem databáze spouští jednotlivé metody pro získání informací, které uživatel zvolil při jejím spuštění.

```
python main.py –N db01 1000 urls.txt –S –H –C –ST –F -dg
```

Zadanými parametry při spuštění aplikace uživatel volí, co bude aplikace o stránce zjišťovat. K tomu slouží tyto metody:

- **get_server** – Tato metoda slouží pro zjištění informace, na jakém serveru stránky „běží“. Této informace je dosaženo analýzou HTTP hlavičky stránky. Vstupem do metody *get_server* je odpověď serveru na dotaz metodou HEAD, která je uložena v databázi. Výstupem je jméno serveru, pokud bylo možné informaci zjistit, případně NaN v případě, že se jednalo o fiktivní název – filtrováno minimální délkou tři znaky. Stejný výstup vrací metoda i v případě, že se informaci nepodařilo získat. K tomu stavu může dojít, pokud server ve své odpovědi nevrací parametr *server*. Na následující straně je hlavní část kódu funkce, která získání informace o serveru zajišťuje. Funkce kontroluje, zda je hodnota parametru *server* v odpovědi přítomna a zda má více znaků než dva. Pokud ano, vrací název serveru, který je na serveru administrátorem vyplněn.

```
...
regex = re.compile('[a-zA-Z-]+')
        reg = regex.search(server)
        if reg != None:
            #ochrana proti fiktivnimu jedno a dvou
pismennemu nazvu serveru
            if len(reg.group(0)) > 2:
                if reg.group(0).lower() in [x.lower()
for x in self.server]:
                    pass
                else:
                    self.server.append(reg.group(0))
                return reg.group(0)
            else:
                return "NaN"
        else:
            return "NaN"

...
```

I přesto, že byla snaha o eliminaci smyšlených názvů serveru (kontrola, zda je název serveru složen alespoň ze tří znaků), na tuto informaci o Word Wide Web stránce je třeba nahlížet spíše jako na doplňující. Administrátor může zadat do názvu serveru jakoukoliv hodnotu. Při činnosti aplikace nebylo výjimkou setkat se se jménem serveru jako Pizza a podobně. Do budoucna by se dalo těmto problémům vyhnout zavedením whitelistu, seznamu přípustných hodnot, ve kterém by například byly pouze Apache, Microsoft-IIS a podobně známé názvy. Ale i to přináší problémy – které servery do něho přesně umístit. Proto byla zvolena strategie ukládat všechny názvy serverů.

- **get_dtd** – Metodou *get_dtd* aplikace získává pro uživatele informace, který značkovací jazyk byl pro tvorbu stránky použit. Metoda pracuje na principu kontroly obsahu elementu DOCTYPE. Každá verze HTML nebo XHTML jazyka má jiný obsah, a právě toho je použito k zjištění informace o dané WWW stránce. Mimo tohoto elementu aplikace sleduje i nové tagy, které se objevily až se značkovacím jazykem HTML 5, z toho důvodu, že v této verzi se zápis elementu razantním způsobem zkrátil a není v něm již uváděna verze a specifikace DTD *(Document Type Definition)*.

```
...
if "//dtd xhtml 1.1//" in self.html.lower():
            return "xhtml 1.1"
if "//dtd xhtml 1.0 strict//" in self.html.lower():
            return "xhtml 1.0 strict"
if "//dtd xhtml 1.0 transitional//" in
self.html.lower():
            return "xhtml 1.0 transitional"
if "<header" in self.html.lower() or "<command" in
self.html.lower() or "<details" in self.html.lower() or
"<dialog" in self.html.lower() or "<summary" in
self.html.lower() or "<figure" in self.html.lower() or
"<time" in self.html.lower() or "<track" in
self.html.lower() or "<output" in self.html.lower() or
"<footer" in self.html.lower() or "<section" in
self.html.lower() or "<audio" in self.html.lower() or
"<video" in self.html.lower() or "<canvas" in
self.html.lower() or "<article" in self.html.lower() or
"<aside" in self.html.lower() or "<bdi" in
self.html.lower() or "<datalist" in self.html.lower() or
"<embed" in self.html.lower() or "<figcaption" in
self.html.lower() or "<hgroup" in self.html.lower() or
"<keygen" in self.html.lower() or "<mark" in
self.html.lower() or "<meter" in self.html.lower() or
"<nav" in self.html.lower() or "<progress" in
self.html.lower() or "<rp" in self.html.lower() or "<rt"
in self.html.lower() or "<ruby" in self.html.lower() or
"<source" in self.html.lower() or "<wbr" in
self.html.lower() or "<meta charset=" in
self.html.lower():
            return "html5"
...
return "NaN"
```

Tělo funkce je mnohem delší, zde jen pro ukázku zjišťování, zda je použit značkovací jazyk XHTML v konkrétní verzi, nebo jazyk HTML ve verzi 5. Vstupem do funkce je HTML, případně XHTML kód stránky, který je převeden na sekvenci malých písmen. V tomto kódu je pak vyhledávána sekvence znaků určující verzi jazyka, nebo dojde k rozhodování na základě tagů. Výstupem je typ a verze značkovacího jazyka, byl-li rozpoznán, případně je návratovou hodnotou z funkce NaN.

- **get_style** – Metoda pro zjištění verze používaných kaskádových stylů *(CSS)* je rozsáhlejší, proto bude popsána pouze slovně bez příkladu kódu. Její kompletní zdrojový kód se nachází v softwarové dokumentaci ke zdrojovým kódům aplikace v příloze této technické zprávy. U kaskádových stylů se rozlišují dvě verze. Jednou z nich je CSS 2.0 a druhou CSS 3.0, která obsahuje nové elementy. Na základě toho je možné rozeznat, zda se jedná o jednu, nebo druhou verzi kaskádových

stylů. Funkce pro zjištění použité verze CSS pracuje jak se zápisem stylů mezi tagy *<head>* a *</head>*, tak i s externím umístěním. Vstupem do metody je zdrojový kód stránky získaný z databáze. Výstupem metody je verze použitého kaskádového stylu. Nebyl-li styl nalezen, je navrácena hodnota NaN.

- **get_scripting** – Tato metoda slouží pro zjištění, jestli byla použita skriptovací technologie na straně serveru, a to konkrétně zda se jednalo o skriptovací jazyk PHP, nebo konkurenční řešení od Microsoftu, ASP.NET. Zjišťování informace probíhá dvěma způsoby. Prvním z nich je využití odpovědi od serveru na dotaz prostřednictvím metody HEAD. Analýzou hlavičky lze zjistit, pokud to server uvádí ve své odpovědi, jaké skriptovací technologie je použito. Jedná se o pole *x-powered-by*. Druhým způsobem, pokud se informace nenacházela již v hlavičce, je prozkoumání zdrojového kódu World Wide Web stránky. Vstupem je celý zdrojový kód stránky, který metoda prochází a vyhledává v něm zmínku o PHP nebo ASP.NET, v závislosti na křížových odkazech.

```
...
if "php" in str_head.lower():
            return "PHP"
        else:
...
```

V ukázce byla zvolena část kódu, která se stará o zjištění informace z hlavičky odpovědi serveru, a to konkrétně, zda byl použit skriptovací jazyk PHP.

- **get_jsf** – Poslední metodou pro vytěžování informací z dat je *get_jsf*. Touto metodou je aplikace schopna zjistit, zda je použit jeden z uvažovaných JavaScript frameworků. V současné době metoda rozpoznává šest frameworků, které byly vybrány na základě určitých kritérií, uvedených v teoretické části této technické zprávy. Jedná se o tyto JavaScript frameworky: jQuery, YUI, Prototype, MooTools, DHTMLX, AngularJS a Dojo.

```
...
regex = re.compile('src="[a-z0-9.:&#\?=/_-]+jquery[a-z0-
9.&#\?=/_-]+\.js')
        reg = regex.findall(self.html.lower())
        if len(reg) > 0:
            return "jQuery"
        else:
            regex = re.compile('src="[a-z0-9.:&#\?=/_-
]+yui[a-z0-9.&#\?=/_-]+\.js')
            reg = regex.findall(self.html.lower())
            if len(reg) > 0:
                return "YUI"
            else:
...
```

Vstupem do metody je (X)HTML kód stránky, který je poskytnut metodě z databáze a přísluší konkrétní stránce, nad kterou jsou prováděny zjišťovací operace. Způsob získání informace o použitém frameworku je postaven na průchodu zdrojovým kódem stránky a pomocí regulárních výrazů obsahujících sekvence znaků typických pro daný framework metoda vrátí název příslušného JavaScript frameworku. Pokud k tvorbě stránky žádného nebylo použito, navrací NaN.

Při vytěžování informací aplikace průběžně generuje SQL skript pro naplnění MySQL databáze nově získanými daty, které uživatele zajímají. V závěru této fáze je vygenerován konfigurační skript, který obsahuje definice pro vytvoření tabulek v této databázi. O struktuře databáze a prezentaci výsledků více v podkapitole *3.5 Prezentace výsledků*.

3.4 Aplikae sPYnet

Aplikace, která v rámci diplomové práce vznikla, nese jméno sPYnet (**s**earch **py**thon app for inter**net**), jedná se o zkratku složenou z počátečních písmen jednotlivých slov, kde první část názvu *sPY* vystihuje, jakou činnost aplikace vykonává, z anglického pozorovat, vyzvídat.

obr. č. 8 – logo aplikace sPYnet

Pro svůj chod aplikace potřebuje interpreter jazyka Python ve verzi 2.7.x, objektově orientovanou databázi Apache CouchDB a knihovnu spycouch pro komunikaci s ní. Databázi lze stáhnout ze stránek CouchDB v podobě zdrojových kódů a pro konkrétní PC ji přeložit a nainstalovat, případně ji získat skrze repozitáře linuxových systémů. Na operačním systému Ubuntu, který byl použit pro testování aplikace, stačí do terminálu zadat příkaz:

```
sudo apt-get install couchdb
```

Instalační proces ověří všechny závislosti, zde se jedná zejména o přítomnost Erlang jazyku. Pokud v systému dosud není, bude uživatel vyzván pro povolení jeho instalace. Instalace proběhne současně s instalací CouchDB.

Po instalaci, aby mohl být databázový stroj vždy při restartu PC sám spuštěn, je třeba mu přidělit práva. K tomu slouží tento příkaz vyvolaný též z okna terminálu:

```
sudo chown -R couchdb /var/run/couchdb
```

Nyní je SŘDB připraven k práci. Pro úplnost dva příkazy pro nucený restart databázového stoje:

```
sudo /etc/init.d/couchdb start

sudo /etc/init.d/couchdb start
```

Spuštění aplikace probíhá z příkazové řádky. První spuštění aplikace obsahuje přepínač **-N**, název databáze, do které budou ukládána data, množství WWW stránek k analýze, cestu k souboru s výchozími URL adresami a přepínače pro jednotlivé metody, které budou použity pro získání informací o stránkách.

```
python main.py -N db01 2300 urls.txt –S –H –C –ST –F -dg
```

- -S – název serveru
- -H – značkovací jazyk
- -C – verze kaskádových stylů
- -ST – skriptovací jazyk na straně serveru
- -F – javaScript framework

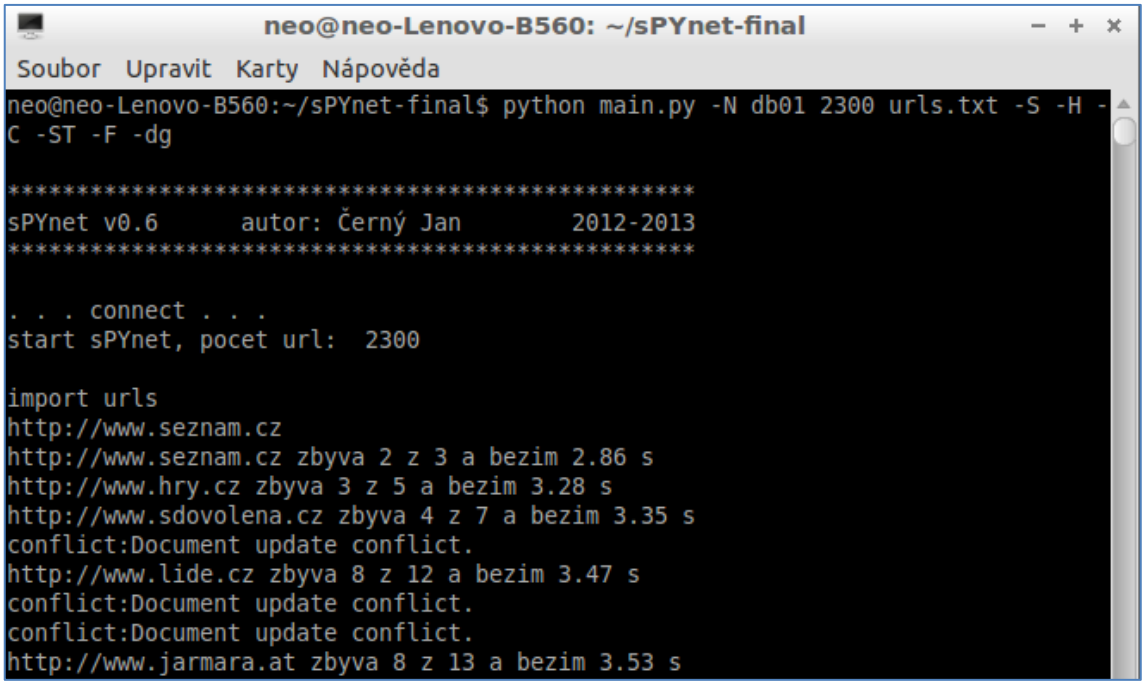
Byla-li aplikace ukončena a uživatel chce dále analyzovat, spouští se znovu příkazem:

```
python main.py -R db01
```

Export URL adres z databáze je spuštěn příkazem:

```
python main.py -E db01
```

Přepínač **-dg** při prvním spuštění určuje, že chce uživatel zobrazovat průběh činnosti do konzolového okna aplikace.



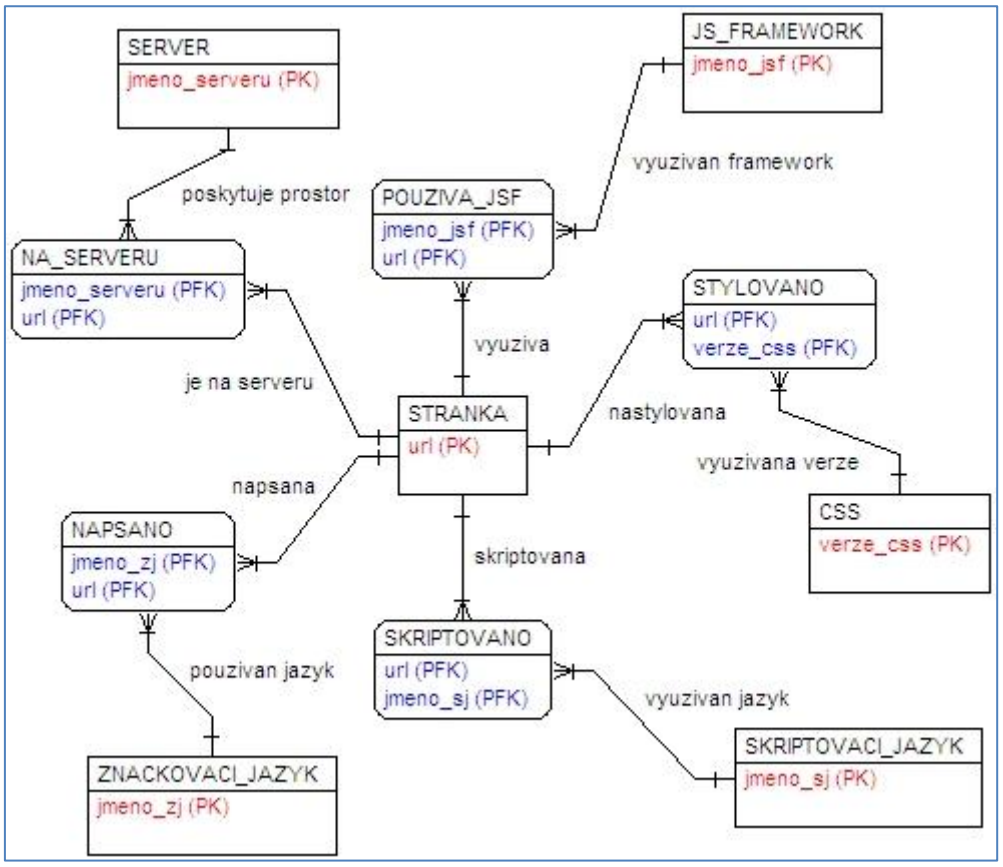
obr. č. 9 – sPYnet v režimu sbírání URL adres

3.5 Prezentace výsledků

Veškeré informace, které o každé stránce aplikace sPYnet získá, jsou přidávány do SQL skriptu, který je spolu s konfiguračním skriptem po skončení činnosti aplikace uživatelem nahrán na databázový MySQL server. Z této databáze, nyní již relační, jsou prezentovány statistické výsledky o World Wide Web stránkách. K prezentaci výsledků v podobě koláčových grafů s legendou slouží PHP skripty, které jsou rovněž přílohou této technické zprávy.

Strukturu relační databáze zachycuje E-R diagram na schématu. Na obrázku *(obr. č. 10 – E-R diagram)* je šest entit a pět vztahů, do kterých mezi sebou vstupují.

Jak entity, tak vztahy jsou v tomto druhu databáze reprezentovány tabulkami. Hlavní tabulkou je tabulka s názvem **STRANKA.** Ta uchovává URL adresy – atribut *url*, který je primárním klíčem, a zároveň jediným atributem této tabulky. Další tabulkou je **SERVER.** Obsahuje rovněž jediný atribut, který je primárním klíčem. V této tabulce jsou uloženy názvy serverů, na kterých jsou WWW stránky dostupné.



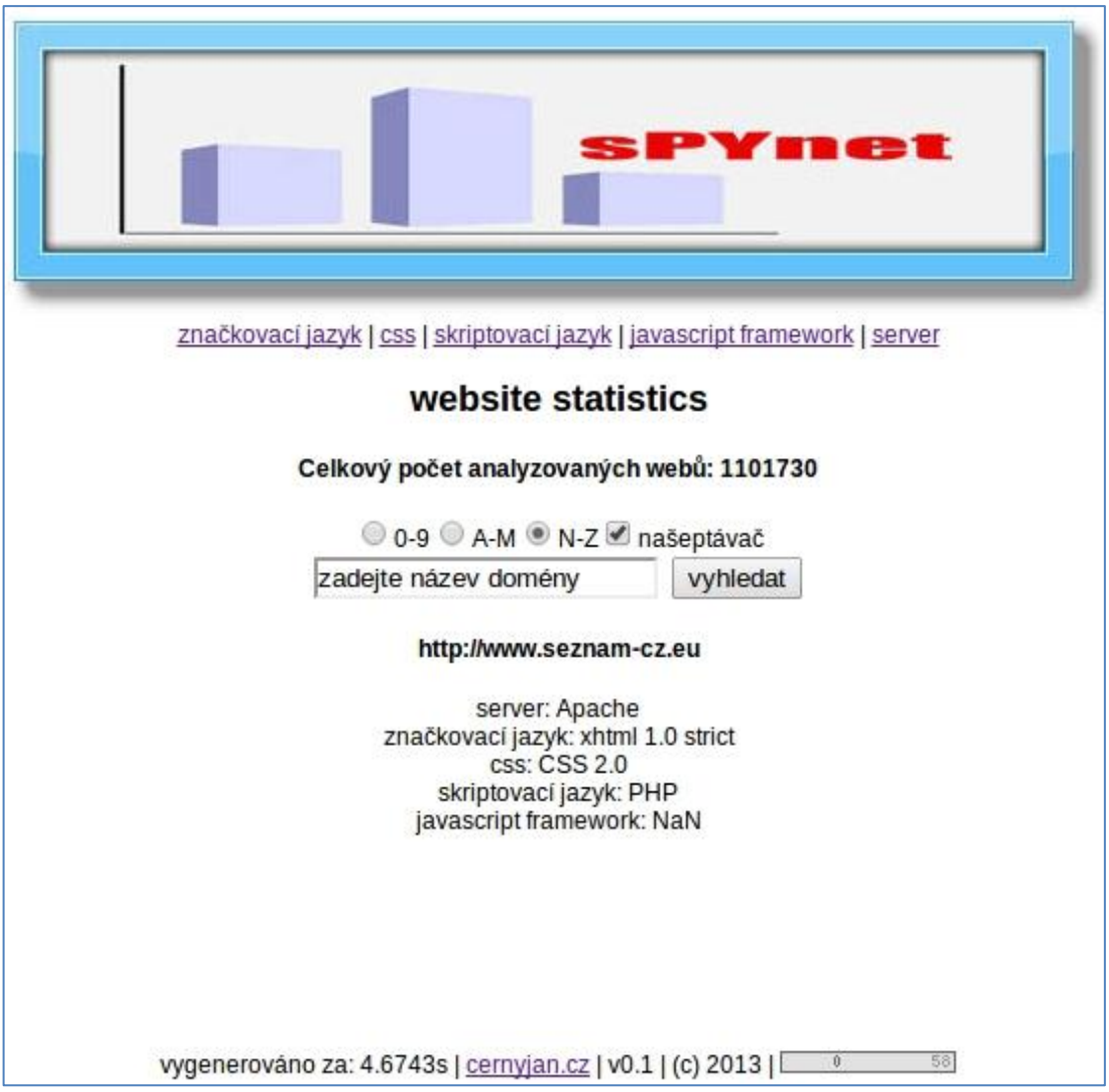
obr. č. 10 – E-R diagram

Tabulka **ZNACKOVACI_JAZYK**, uchovává názvy a verze jednotlivých značkovacích jazyků, které jsou používány k tvorbě World Wide Web stránek. V tabulce **CSS** jsou uloženy verze kaskádových stylů. Tabulka **SKRIPTOVACI_JAZYK** v sobě uchovává názvy skriptovacích jazyků pro skriptování na straně serveru. Poslední tabulkou s jediným atributem je tabulka s názvem **JS_FRAMEWORK,** do které jsou uloženy názvy JavaScript frameworků. Další tabulkou databáze je **NA_SERVERU**, která má dva atributy, oba dva jsou cizími klíči. Tabulka uchovává hodnoty URL adresa a jméno serveru, na kterém sídlí. Tabulka s názvem **NAPSANO** obsahuje hodnoty URL adresa a název s verzí značkovacího jazyka použitého při tvorbě WWW stránky. **STYLOVANO** je další tabulka, ve které jsou hodnoty URL adresa a verze

CSS. Tabulka **SKRIPTOVANO** uchovává URL adresu a název skriptovacího jazyka pro skriptování na straně serveru. Poslední tabulka s názvem **POUZIVA_JSF** slouží pro ukládání URL adresy a jména použitého JavaScript frameworku.

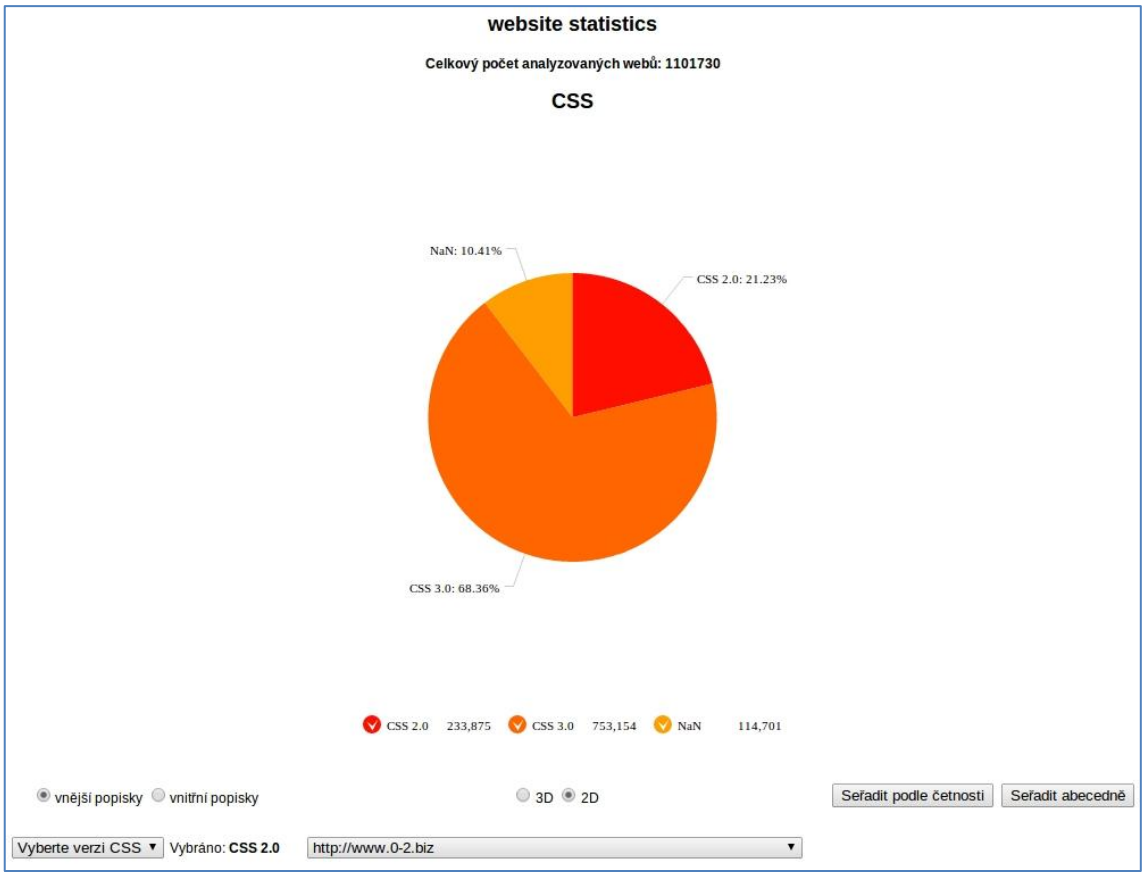
Webová stránka, jejíž formou jsou statistiky prezentovány, na své index stránce obsahuje formulář pro vyhledávání konkrétní stránky, o které chce uživatel získat informace. Formulář obsahuje „našeptávač“, který uživateli slouží pro rychlé vyhledání jeho požadované WWW stránky. S postupným zadáváním znaků se výsledek dotazu zpřesňuje. Tuto funkcionalitu lze vypnout, zatrhnutím checkboxu. Tato volba je pak uchována po celou dobu sezení. Pro rozložení zátěže serveru jsou analyzované stránky v „našeptávači“ rozděleny do tří sekcí. První z nich shromažďuje World Wide Web stránky, na které odkazuje URL adresa začínající číslem, do druhé sekce patří URL adresy od A po M, poslední sekce obsahuje zbytek URL adres (od N do Z).

Na následující straně je na obrázku *(obr. č. 11 – úvodní stránka)* zobrazena index stránka, na které uživatel vyhledal pomocí interaktivního formuláře informace pro World Wide Web stránku umístěnou na URL adrese *http://www.seznam-cz.eu*. Dostal odpověď, že tato stránka „běží“ na serveru Apache, pro její tvorbu bylo použito značkovacího jazyka XHTML ve verzi 1.0 Strict. Dále získá informaci o tom, že je na stránce použito kaskádových stylů, a to konkrétně verze 2.0. Poslední dostupnou informací je, že stránka byla vygenerována za pomoci skriptovací technologie na straně serveru. Tímto skriptovacím jazykem bylo PHP. Přítomnost jednoho z předdefinovaných JavaScript frameworků nebyla potvrzena.



obr. č. 11 – úvodní stránka

Součástí stránky pro podporu diplomové práce je rozcestník, který vede na jednotlivé zkoumané informace. Každý zkoumaný parametr World Wide Web stránek je dostupný z podstránky, která obsahuje graf a legendu. Dále se na stránce nacházejí funkční prvky jako přepínače a „roletové“ nabídky. Přepínače slouží pro způsob zobrazení popisků grafu. Uživatel má na výběr ze dvou způsobů – vnější a vnitřní popisky. Druhým přepínačem se mění zobrazení grafu. Graf může být 2D, jak je zobrazeno na obrázku *(obr. č. 12 – výsledky pro CSS)*, případně je přidán i třetí rozměr a graf se jeví jako 3D objekt.



obr. č. 12 – výsledky pro CSS

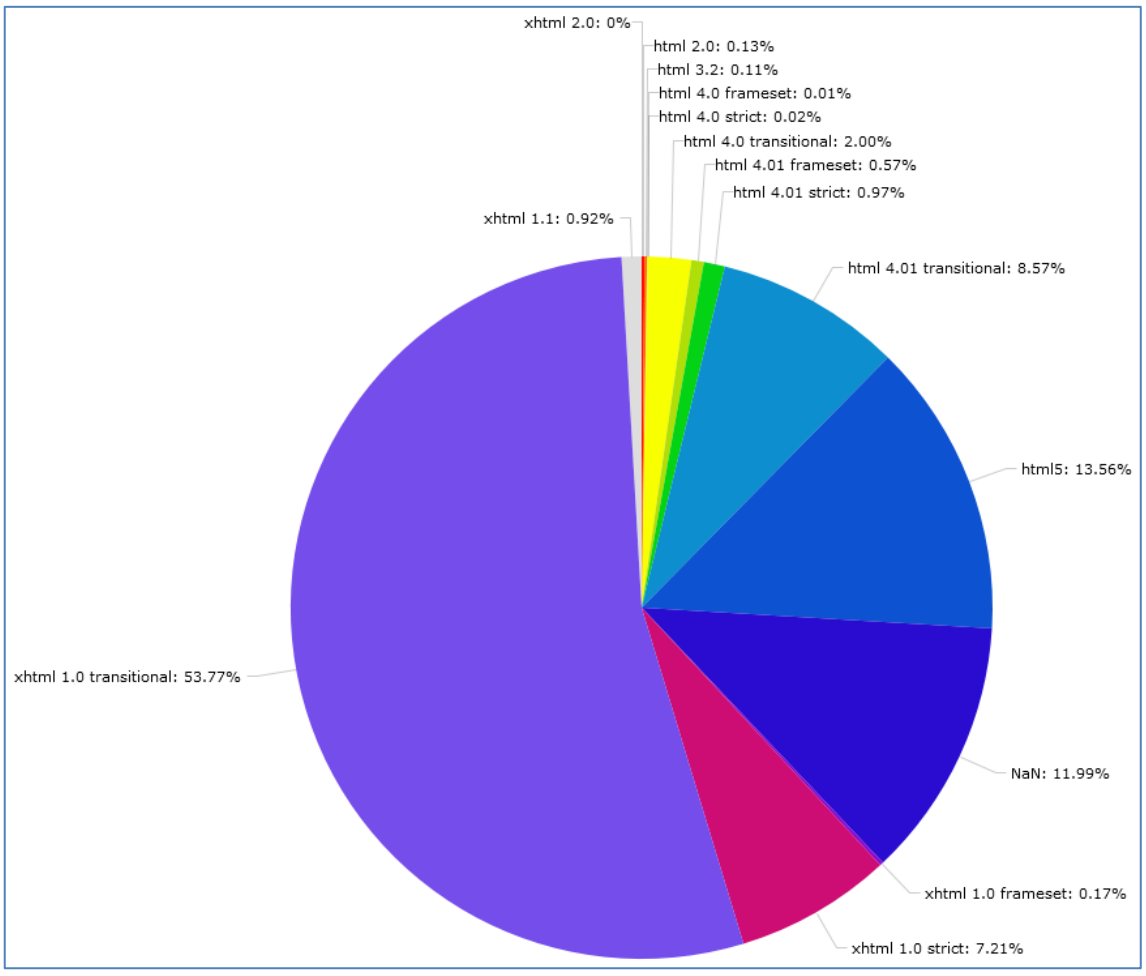
Informace lze řadit dle abecedy, nebo četnosti jejího výskytu. Další funkcí je možnost některé položky v grafu potlačit. Uživatel si tak může zvolit, co se mu bude konkrétně zobrazovat – kliknutím na příslušnou položku legendy. Opětovné kliknutí na ni ji znovu přidá do výběru. V dolní části stránky se nachází roletová nabídka, která slouží pro výběr konkrétní informace zjištěné o WWW stránkách. Pokud položku uživatel zvolí, automaticky se mu druhá nabídka předvyplní těmi stránkami, u kterých byla zjištěna. Tím je zajištěno i vyhledávání „z druhé strany“, kdy uživatel nevolí konkrétní stránku, o které se chce dozvědět informace, ale na základě konkrétní informace získá přehled stránek, u kterých byla zjištěna.

Pro grafovou reprezentaci výsledů bylo použito zdarma dostupné varianty JavaScriptové knihovny s názvem *amCharts*. Její použití je podmíněno zobrazením malého odkazu na domovskou stránku u každého grafu, které nijak neruší.

3.6 Výsledky a vyhodnocení

Obsahem této podkapitoly jsou statistické výsledky analýzy WWW stránek. Zadáním diplomové práce bylo určeno, že má být zanalyzován nejméně jeden milion WWW stránek. Pomocí aplikace sPYnet bylo zanalyzováno přesně **1 200 035** unikátních World Wide Web stránek. Konečné číslo, ze kterého jsou provedena statistická měření, je 1 101 730 unikátních WWW stránek sídlících na doménách II. řádu. V době měření na 98 305 dotazů nedostala v předem stanoveném časovém rámci (proměnná timeout nastavena na 2s) aplikace odpověď. Tyto stránky proto nejsou zahrnuty v testovaném vzorku. Hodnoty v procentech jsou zaokrouhlovány na 2 desetinná místa.

3.6.1 Značkovací jazyk



obr. č. 13 – značkovací jazyk

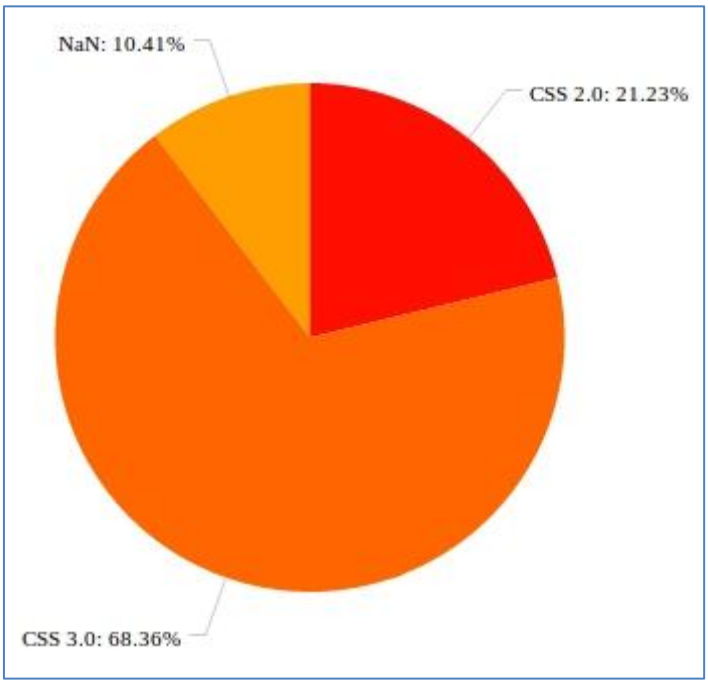
Z testovaného vzorku WWW stránek jich bylo napsáno 592 413 v XHTML 1.0 Transitional, to je 53,77% zastoupení. Nejméně stánek bylo napsáno ve značkovacím jazyku XHTML ve verzi 2.0, a to rovných 5.

Z přehledu je patrné, že značkovací jazyk HTML ve verzi 5 je druhým nejvíce používaným jazykem při tvorbě hypertextových dokumentů ve World Wide Webu. Navzdory tomu, že není dosud oficiálně standardizován (r. 2013).

tab. č. 2 – značkovací jazyky

Název	počet	procentuální zastoupení
HTML 2.0	1 447	0,13%
HTML 3.2	1 251	0,11%
HTML 4.0 Frameset	120	0,01%
HTML 4.0 Strict	251	0,02%
HTML 4.0 Transitional	22 012	2,00%
HTML 4.01 Frameset	6 287	0,57%
HTML 4.01 Strict	10 642	0,97%
HTML 4.01 Transitional	94 460	8,57%
HTML5	149 344	13,56%
NaN	132 095	11,99%
XHTML 1.0 Frameset	1 818	0,17%
XHTML 1.0 Strict	79 483	7,21%
XHTML 1.0 Transitional	592 413	53,77%
XHTML 1.1	10 102	0,92%
XHTML 2.0	5	0%

3.6.2 Kaskádové styly



obr. č. 14 – CSS

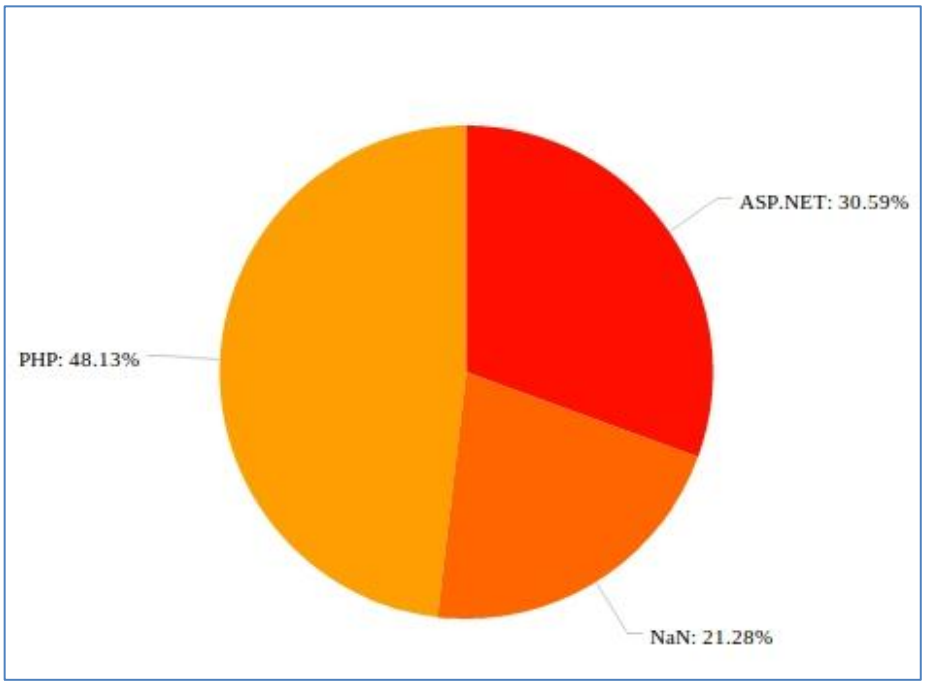
Z testovaného vzorku World Wide Web stránek používalo rovných 753 154 stránek kaskádové styly ve verzi 3.0, jedná se o 68,36% zastoupení této verze. Starší verzí CSS 2.0 bylo „nastylováno“ 233 875 unikátních stránek.

Přibližně u jedné desetiny testovaných stránek nebyla přítomnost žádné verze kaskádových stylů potvrzena. Na této skutečnosti se podílí i to, že starších verzí jazyka HTML (verze 2.0 až 4.01), které CSS nepodporují, je stále v jisté míře k tvorbě WWW stránek využíváno.

tab. č. 3 – kaskádové styly

název	počet	procentuální zastoupení
CSS 2.0	233 875	21,23%
CSS 3.0	753 154	68,36%
NaN	114 701	10,41%

3.6.3 Skriptovací jazyk



obr. č. 15 – skriptovací jazyk

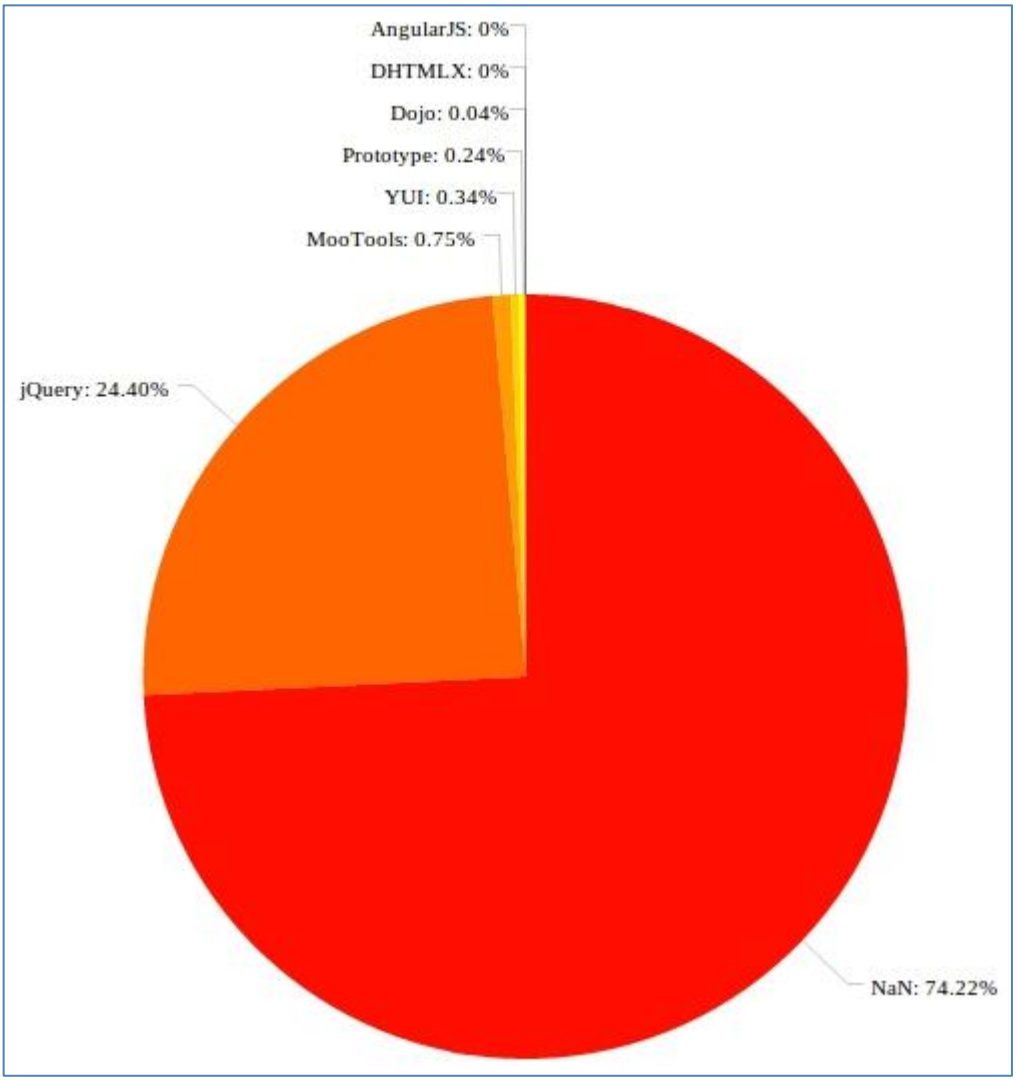
Z testovaného vzorku stránek jich bylo 530 288 vytvořeno za pomoci skriptovací technologie jazyka PHP. Toto číslo se rovná 48,13% zastoupení ze všech analyzovaných World Wide Web stránek, které byly součástí zkoumaného vzorku. Technologií ASP.NET bylo vytvořeno 337 024 stránek. Stránky, na které nebylo dle statistik použito ani jedné z těchto technologií tvořily 21,28%. Jazyk nutně nemusí zanechat stopu, podle které lze určit, zda byla stránka pomocí něho vytvořena.

Pokud by bylo na počet stránek využívajících PHP nahlíženo z širšího hlediska a bylo by vzato v úvahu spojení PHP-Apaches *(Apache, nejčetnější server z tab. č. 6 – servery)* je tento výsledek logickým vyústěním. Výsledky rovněž potvrzují, že PHP je nejrozšířenější. Tuto skutečnost tvrdí i řada dalších statistik, například výsledky z obdobného projektu *W3Techs* (srpen 2012).

tab. č. 4 – skriptovací jazyk

název	počet	procentuální zastoupení
ASP.NET	**337 024**	**30,59%**
PHP	**530 288**	**48,13%**
NaN	**234 418**	**21,28%**

3.6.4 JavaScript framework



obr. č. 16 – javaScript framework

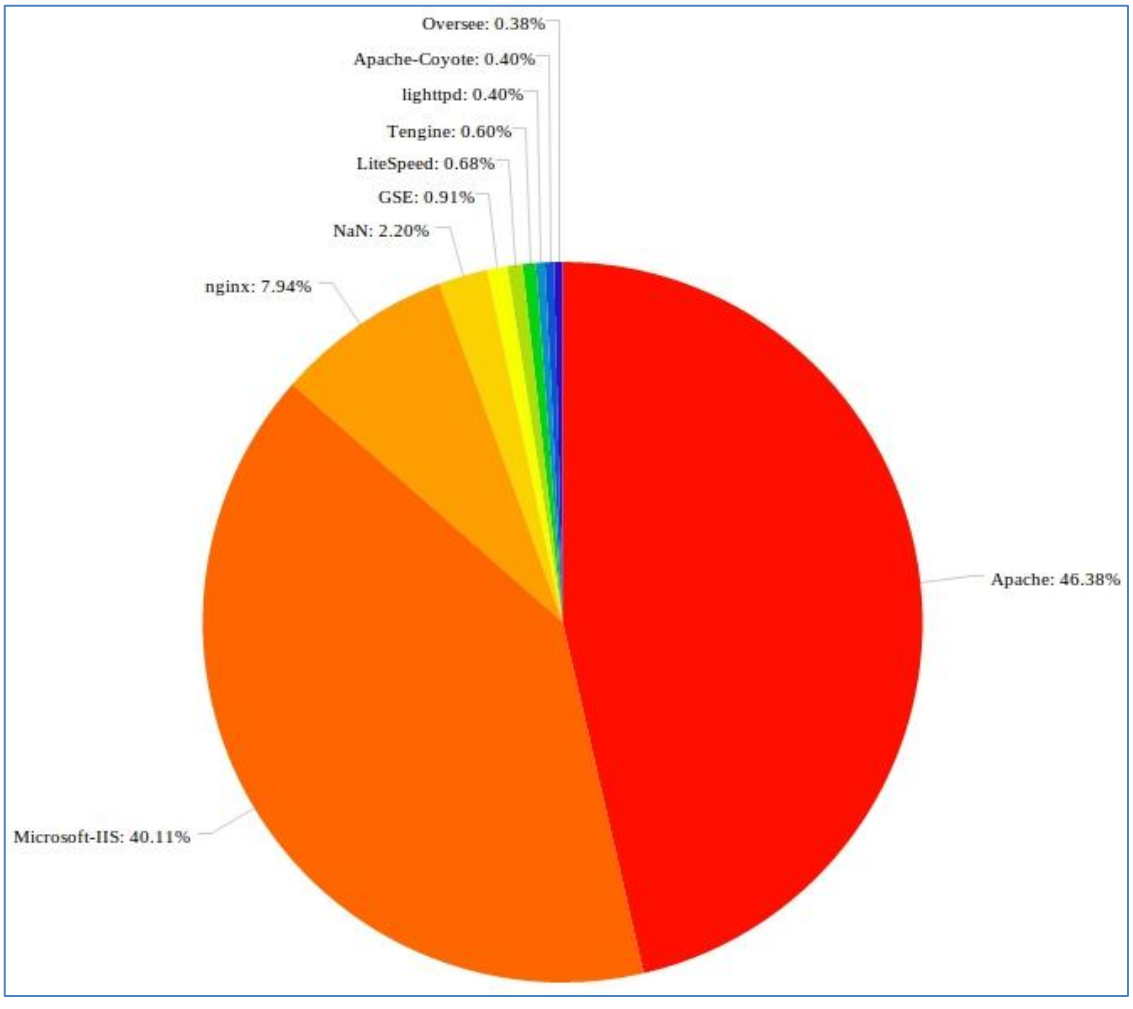
Nejvíce WWW stránek využívalo jQuery framework – 268 796 stránek. To je 24,40% zastoupení z testovaného vzorku. Tento výsledek potvrzuje, že jQuery je nejrozšířenějším JavaScript frameworkem.[20] Stránek, u kterých nebyla přítomnost jednoho ze sedmi předem vybraných JavaScript frameworků nalezena, bylo 817 750. Toto číslo nevypovídá o tom, kolik stránek v globálním měřítku žádný JavaScript framework nepoužívá, ale že nepoužívá konkrétní z výběru.

Zajímavým zjištěním je, že třetím nejpoužívanějším frameworkem je YUI, který nebyl uveden ani v jednom z článků zmíněných v kapitole *2 Teoretická část*. Ostatní výsledky z analýzy stránek na přítomnost vybraných JavaScript frameworků zobrazuje tabulka *(tab. č. 5 – javaScript framework)* na následující straně.

tab. č. 5 – javaScript framework

název	počet	procentuální zastoupení
AngularJS	2	0%
DHTMLX	31	0%
Dojo	398	0,04%
jQuery	268 796	24,40%
MooTools	8 313	0,75%
NaN	817 750	74,22%
Prototype	2 651	0,24%
YUI	3 789	0,34%

3.6.5 Server



obr. č. 17 – server

Předložit výsledky z analýzy WWW stránek, která se zaměřovala na získání informace, na kterém serveru stránky „běží“, by několikanásobně překročily stránkový rozsah práce. Proto kompletní výsledky nejsou ani součástí příloh. Jsou pouze dostupné z webové stránky *http://www.cernyjan.cz/diplomka/statistiky-server.php*, která slouží pro podporu této diplomové práce. V této technické zprávě k diplomové práci je vybráno z celkového počtu všech získaných dat 10 serverů s největším zastoupením. Procentuální zastoupení zobrazuje graf na předešlé straně a aktuální počet WWW stránek „běžících“ na konkrétním serveru je pod textem v tabulce *(tab. č. 6 – servery)*.

tab. č. 6 – servery

název	počet	procentuální zastoupení
Apache	**498 548**	**46,38%**
Microsoft-IIS	**431 124**	**40,11%**
nginx	**85 402**	**7,94%**
NaN	**23 618**	**2,20%**
GSE	**9 748**	**0,91%**
LiteSpeed	**7 345**	**0,68%**
Tengine	**6 407**	**0,60%**
lighttpd	**4 345**	**0,40%**
Apache-Coyote	**4 314**	**0,40%**
Oversee	**4 103**	**0,38%**

Tabulka zobrazuje, že *nginx* je po nejznámějších dvou zástupcích webových serverů třetím nejvíce dotazovaným serverem. Jedná se o reverzní proxy server. Z toho vyplývá, že je snaha na WWW určitým způsobem zajistit rychlou distribuci statického obsahu a rozložení zátěže.

4 Zhodnocení

Doba potřebná k uskutečnění analýzy příslušného vzorku World Wide Web stránek je rozdělena do tří částí. Sběr unikátních URL adres, indexace WWW stránek a samotná analýza. Celková doba, kterou aplikace potřebovala k získání výsledů, které jsou předmětem této diplomové práce, zachycuje tabulka *(tab. č. 5 – doba potřebná pro analýzu)*.

tab. č. 7 – doba potřebná pro analýzu

operace	začátek	konec
sběr adres	**09. 04. 2013**	**16. 04. 2013**
indexace stránek	**17. 04. 2013**	**18. 04. 2013**
analýza (všechny moduly aktivní)	**19. 04. 2013**	**23. 04. 2013**

Drobné problémy, se kterými se autor práce setkal, byly postupně zmíněny v jednotlivých kapitolách. Jeden významný problém se však vyskytl až v průběhu finálního testování aplikace. Původní výběr objektově orientované databáze – Apache CouchDB, který se zdál jako nejlepší volbou, se ukázal při nejmenším jako značně komplikující realizaci navrhované aplikace. Problém se skrýval v knihovnách pro její obsluhu, které poskytují třetí strany (CouchDB své vlastní řešení pro jednotlivé jazyky nenabízí). V průběhu práce byla aktuální verze CouchDB 1.0.1 a poté 1.2.1. Všechny dostupné knihovny úspěšně podporovaly databázi až do verze 0.9.x. Zvolená knihovna *couchdb-python* i bez větších potíží spolupracovala s verzí 1.0.1, ale při updatu databáze na vyšší verzi přestala fungovat.

V průběhu práce přecházet na úplně novou databázi od jiných tvůrců, kde by se mohl časem objevit stejný problém, nebylo vhodné, proto byla vytvořena jednoduchá python knihovna s názvem *spycouch* (Simple Python API for CouchDB), která řeší problém s nekompatibilitou dostupných knihoven v jazyku Python pro obsluhu CouchDB databázového serveru.

4.1 spycouch

Jednoduché python API pro Apache CouchDB. Na rozdíl od běžně dostupných knihoven, které pracují s databází do verze 0.9.x, tato pracuje i s aktuálně dostupnou verzí 1.2.1. Mezi její funkce patří:

Operace nad databází

- vytvoření nové databáze na serveru
- smazání databáze
- výpis všech dostupných databází, které CouchDB spravuje
- získání informací o jednotlivé databázi
- komprese dat v databázi
- vytváření a používání map pohledů

Operace s dokumenty

- výpis všech dostupných dokumentů v databázi
- získání obsahu dokumentu
- uložení/editace dokumentu
- smazání dokumentu z databáze

Knihovna je dostupná z: *https://github.com/cernyjan/repository/tree/master/spycouch*

5 Závěr

Výsledkem diplomové práce s názvem Statistika WWW stránek je plně funkční aplikace v programovacím jazyku Python. Aplikace slouží pro automatickou analýzu World Wide Web stránek, pomocí níž lze databázi analyzovaných WWW stránek rozšířit a získat tak větší testovaný vzorek, případně opětovně testovat již analyzované stránky a zkoumat míru změn v čase. Mimo aplikace obsahuje přiložený DVD nosič, databázi výsledků měření a seznam analyzovaných webů.

Dalším přínosem kromě aplikace samotné a výsledků analýzy je i python balíček – API pro obsluhu Apache CouchDB databáze z prostředí programovacího jazyku Python, který vznikl při řešení diplomové práce.

Na aplikaci je možné dále stavět a vzhledem k její modularitě ji lze doplnit o další moduly, které by rozšířily její stávající funkcionalitu – získávání dalších informací o World Wide Web stránkách.

Citace a použité zdroje

[1] RFC 2616. *Hypertext Transfer Protocol*. W3C: The Internet Society, 1999. Dostupné z: http://www.w3.org/Protocols/rfc2616/rfc2616.html

[2] REC-xhtml1. *XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition)*. W3C: MIT, 2002. Dostupné z: http://www.w3.org/TR/2002/REC-xhtml1-20020801/

[3] *Úvod do XML* [online]. 2010 [cit. 2013-05-07]. Dostupné z: http://www.nti.tul.cz/~satrapa/vyuka/xml/prednaska01.pdf

[4] The PHP License, version 3.01. *PHP License*. The PHP Group, 2012. Dostupné z: http://www.php.net/license/3_01.txt

[5] BLAHOUT, Michal. Co je ASP?. *Jemný úvod do ASP* [online]. 2002 [cit. 2013-05-07]. Dostupné z: http://www12.brinkster.com/mibla/co_je_asp.asp

[6] POCHYLA, Martin. *JavaScript* [pdf]. VŠB – Technická univerzita Ostrava, 2006, 96 s. [cit. 2013-05-07].

[7] Speckyboy. ANDREW, Paul. *Top 10 Javascript Frameworks – Which do you prefer?* [online]. 2008 [cit. 2013-05-07]. Dostupné z: http://speckyboy.com/2008/04/01/top-10-javascript-frameworks-which-do-you-prefer/

[8] VIKAS. Zoomzum. *10 Javascript Framework/Libraries for Developers* [online]. 2012 [cit. 2013-05-07]. Dostupné z: http://zoomzum.com/10-javascript-frameworklibraries-for-developers/

[9] ŠVEC, Jan. *Létající cirkus: Python tutoriál* [pdf]. 2003, 119 s. [cit. 2013-05-07].

[10] PALOVSKÁ, Helena. Datové modelování. *Krokodýlovy databáze* [online]. 2011 [cit. 2013-05-07]. Dostupné z: http://krokodata.vse.cz/DM/DM

[11] Úvod do JSON. *Introducing JSON* [online]. 2012 [cit. 2013-05-07]. Dostupné z: http://www.json.org/json-cz.html

[12] Apache License, Version 2.0. *Apache License*. The Apache Software Foundation, 2004. Dostupné z: http://www.apache.org/licenses/LICENSE-2.0.html

[13] Erlang. *IT Slovník* [online]. 2010 [cit. 2013-05-07]. Dostupné z: http://it-slovnik.cz/pojem/erlang

[14] BSON. *MongoDB* [online]. 2012 [cit. 2013-05-07]. Dostupné z: http://docs.mongodb.org/meta-driver/latest/legacy/bson/

[15] KENKRE, Poonam. *Web Crawlers* [online]. 2012, 43 s. [cit. 2013-05-14]. Dostupné z: http://www.slideshare.net/poonamkenkre/web-crawler-14590800

[16] CASTILLO, C. *Scheduling Algorithms for Web Crawling* [online]. 2007, 50 s. Dostupné z: http://www.slideshare.net/ChaToX/scheduling-algorithms-for-web-crawling

[17] PORT – Spojení s vědou ze všech stran. *Datamining – dolování dat* [online]. 2007 [cit. 2013-05-07]. Dostupné z: http://www.ceskatelevize.cz/program/port/125-datamining-dolovani-dat/

[18] LIU, Bing. Web data mining: exploring hyperlinks, contents, and usage data. Berlin: Springer, 2007, xix, 532 s. ISBN 978-3-540-37881-5.

[19] ČIČÁK, Matěj. Google Trends: O co se lidé zajímají?. *Jak na počítač* [online]. 2012 [cit. 2013-05-07]. Dostupné z: http://jnp.zive.cz/google-trends-o-co-se-lide-zajimaji

[20] HASSMAN, Martin. PHP používají tři čtvrtiny webů, jQuery polovina. Plus další statistiky z W3techs. *Zdroják: Různé* [online]. 2012, č. 1 [cit. 2013-05-15]. Dostupné z: http://www.zdrojak.cz/zpravicky/php-pouzivaji-tri-ctvrtiny-webu-jquery-polovina-plus-dalsi-statistiky-z-w3techs/

Seznam příloh

Příloha A

Softwarová dokumentace

sPYnet

Package sPYnet-final :: Module main'

Module main'

name: sPYnet -- website statistics autor: Černý Jan email: cerny.jan@hotmail.com

version: 0.6

Classes	
	Couch Class of Simple Python API for CouchDB
	Methods Zpracovani dokumentu a ziskani hledanych udaju
	Thread

Functions	
string	**change_http_format**(url) Zmena formatu protokolu pro CouchDB
bool	**check_domain**(url) Kontrolni dotaz smerovany na server o pozadovanou stranku - metoda HEAD
bool	**check_url**(url) Kontrola, zda je URL v pozadovanem tvaru
	exit(status=...) Exit the interpreter by raising SystemExit(status).
array	**extract_urls**(string) Vyzobnuti URL adres ze stranky
	generate_sql(server,operations, db_name,parameters, sql_db) Generovani sql skriptu pro prezentaci vysledku na webu
string	**get_charset**(html, head) Zjisteni v jake znakove sade je stranka kodovana, pokud nezjisteno

	defaultne se voli utf-8
string	**get_encode**(string, url) Zpracovani stranky pythonem v jejim nativnim kodovani, pripadne v utf-8
string	**get_top_domain**(url) Z URL adresy odfiltrovat pouze Top domenu (TLD) a domenu 2.
array	**get_top_domains**(urls) Z URL adres odfiltrovat pouze Top domeny (TLD) s domenou 2.
array	**http_get**(url) Zadost serveru o pozadovanou stranku - metoda GET
	indexing(url, server,db_name, parameters, grab) Plneni DB url adresami/indexovani stranek - v zavislosti na prepinaci 'grab' Funkce je z procesu spoustena jako samostatne vlakno.
	main() Hlavni proces sPYnet
	processing(url, server,operations, db_name,parameters, sql_db) Operace provadene nad mnozinou dokumentu - vysledky generovany do sql skriptu pro jejich prezentaci na webu Funkce je z procesu spoustena jako samostatne vlakno.
	sleep(seconds) Delay execution for a given number of seconds.
	test_connection() Kontrola funkcniho pripojeni do Internetu dotaz na 'http://www.seznam.cz' pri neuspechu opetovny pokus po 10s
	urlparse(url, scheme='',allow_fragments=True) Parse a URL into 6 components: <scheme>://<netloc>/<path>;<params>?<query>#<fragment> Return a 6-tuple: (scheme, netloc, path, params, query, fragment).
	write_to_log(string) Zapis informace s prislusnym casem do logu

Variables

	__package__ = 'sPYnet-final'
	argv = ['(imported)']

Function Details

***change_http_format*(*url*)**

Zmena formatu protokolu pro CouchDB

Parameters:

- **url** (string) - Uniform Resource Locator.

Returns: string

Zmeneny format protokolu.

***check_domain*(*url*)**

Kontrolni dotaz smerovany na server o pozadovanou stranku - metoda HEAD

Parameters:

- **url** (string) - Uniform Resource Locator.

Returns: bool

True or False.

***check_url*(*url*)**

Kontrola, zda je URL v pozadovanem tvaru

Parameters:

- **url** (string) - Uniform Resource Locator.

Returns: bool

True or False.

***exit*(*status*=...)**

Exit the interpreter by raising SystemExit(status). If the status is omitted or None, it defaults to zero (i.e., success). If the status is numeric, it will be used as the system exit status. If it is another kind of object, it will be printed and the system exit status will be one (i.e., failure).

extract_urls(string)

Vyzobnuti URL adres ze stranky
Parameters:

- **`string`** (string) - HTML kod stranky.

Returns: array

List url adres.

generate_sql(server, operations,db_name, parameters, sql_db)

Generovani sql skriptu pro prezentaci vysledku na webu
Parameters:

- **`server`** (object) - Instance tridy Couch.
- **`operations`** (object) - Instance tridy Methods.
- **`db_name`** (string) - Jmeno DB se kterou se prave pracuje.
- **`parameters`** (array) - List prepinacu, co na strance analyzovat.
- **`sql_db`** (string) - Jmeno DB pro kterou bude generovan sql skript. '

get_charset(html, head)

Zjisteni v jake znakove sade je stranka kodovana, pokud nezjisteno defaultne se voli utf-8
Parameters:

- **`html`** (string) - HTML kod stranky.

Returns: string

Kodovani stranky.

get_encode(string, url)

Zpracovani stranky pythonem v jejim nativnim kodovani, pripadne v utf-8
Parameters:

- **`string`** (string) - Kodovani stranky.

Returns: string

Python format kodovani.

get_top_domain(url)

Z URL adresy odfiltrovat pouze Top domenu (TLD) a domenu 2. radu
Parameters:

- **`url`** (string) - Uniform Resource Locator.

Returns: string

Retezec s odfiltrovanou url.

get_top_domains(urls)

Z URL adres odfiltrovat pouze Top domeny (TLD) s domenou 2. radu
Parameters:

- **`urls`** (array) - List url adres.

Returns: array

List odfiltrovanych url adres.

http_get(url)

Zadost serveru o pozadovanou stranku - metoda GET
Parameters:

- **`url`** (string) - Uniform Resource Locator.

Returns: array

HTML stránka array[0] -> head, array[1] -> body.

indexing(url, server, db_name,parameters, grab)

Plneni DB url adresami/indexovani stranek - v zavislosti na prepinaci 'grab Funkce je z procesu spoustena jako samostatne vlakno.
Parameters:

- **`url`** (string) - Uniform Resource Locator.
- **`server`** (object) - Instance tridy Couch.
- **`db_name`** (string) - Jmeno DB se kterou se prave pracuje.
- **`parameters`** (array) - List prepinacu, co na strance analyzovat.
- **`grab`** (bool) - Prepinac mezi sberem URL adres a indexaci webu. '

***processing*(*url*, *server*, *operations*,*db_name*, *parameters*, *sql_db*)** Operace provadene nad mnozinou dokumentu - vysledky generovany do sql skriptu pro jejich prezentaci na webu Funkce je z procesu spoustena jako samostatne vlakno. **Parameters:** • **`url`** (string) - Uniform Resource Locator. • **`server`** (object) - Instance tridy Couch. • **`operations`** (object) - Instance tridy Methods. • **`db_name`** (string) - Jmeno DB se kterou se prave pracuje. • **`parameters`** (array) - List prepinacu, co na strance analyzovat. • **`sql_db`** (string) - Jmeno DB pro kterou bude generovan sql skript. '
***sleep*(*seconds*)** Delay execution for a given number of seconds. The argument may be a floating point number for subsecond precision.
***urlparse*(*url*, *scheme*=`''`,*allow_fragments*=*True*)** Parse a URL into 6 components: <scheme>://<netloc>/<path>;<params>?<query>#<fragment> Return a 6-tuple: (scheme, netloc, path, params, query, fragment). Note that we don't break the components up in smaller bits (e.g. netloc is a single string) and we don't expand % escapes.
***write_to_log*(*string*)** Zapis informace s prislusnym casem do logu **Parameters:** • **`string`** (string) - Text, ktery ma byt zapsan do logu.

Package sPYnet-final :: Module methods

Module methods

name: Komponenta pro analyzu webu aplikace sPYnet -- website statistics autor: Černý Jan email: cerny.jan@hotmail.com version: 0.6

Classes	
	`Methods` Zpracovani dokumentu a ziskani hledanych udaju

Variables	
	`__package__` = `'sPYnet-final'`

spycouch

Package sPYnet-final :: Package spycouch

Package spycouch

name: Simple Python API for CouchDB autor: Černý Jan email:

cerny.jan@hotmail.com version: 0.1

license: viz. LICENSE

Classes	
	Couch Class of Simple Python API for CouchDB

Variables	
	__package__ = 'sPYnet-final.spycouch'

Příloha B

DVD-ROM

Obsah DVD nosiče

Na disku je kompletní projekt s aplikací sPYnet a knihovnou spycouch z vývojového prostředí Eric python IDE a softwarová dokumentace ke všem zdrojovým kódům ve formě hypertextového dokumentu. Dále se na médiu nachází export URL adres analyzovaných stránek a obsah databáze s výsledky analýzy. Na DVD je i elektronická podoba vytištěné technické zprávy.